$4.95

HYPERLINK MAGAZINE

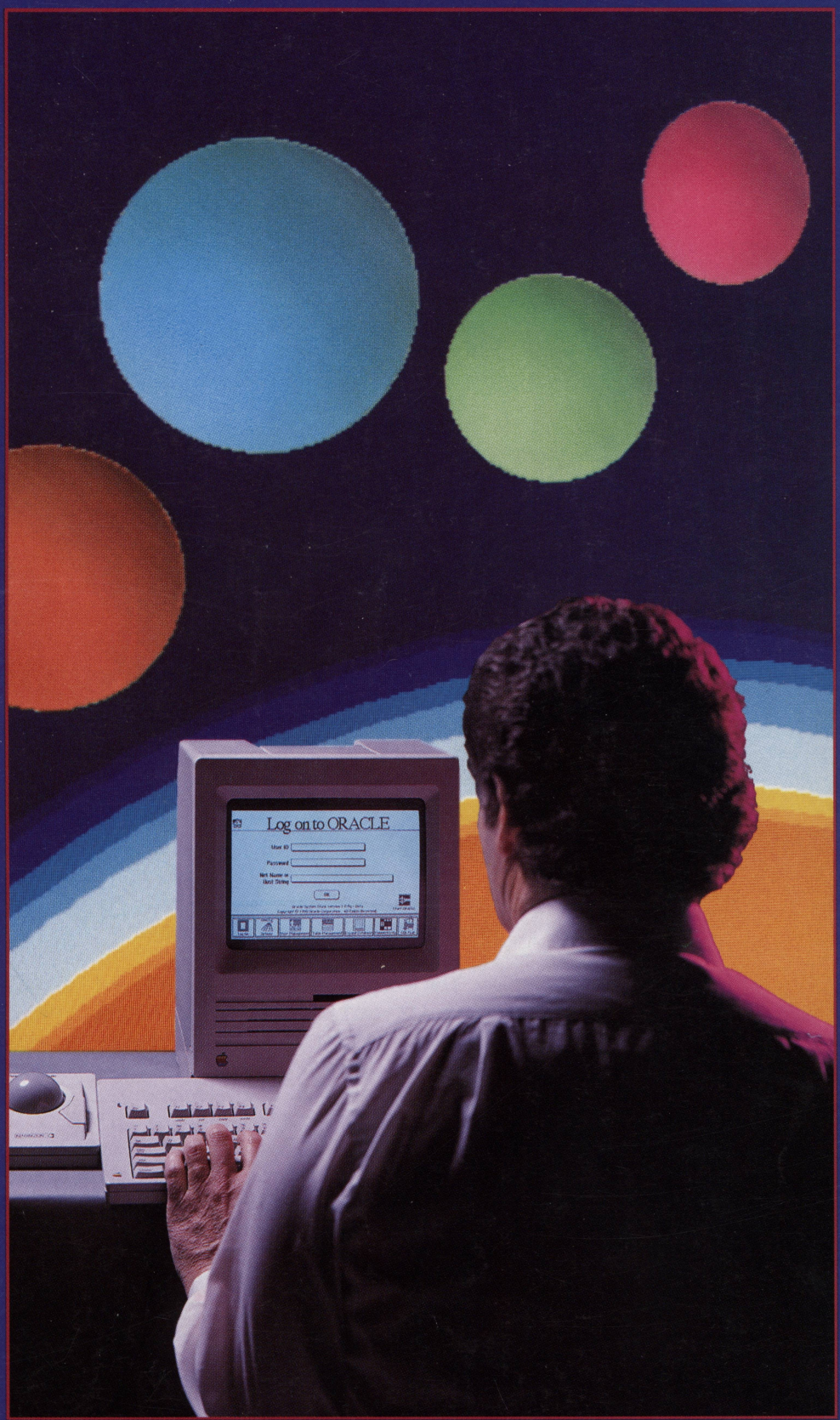
Jan/Feb 1989 • Vol. 2 No. 1

Linking HyperCard To Other Worlds

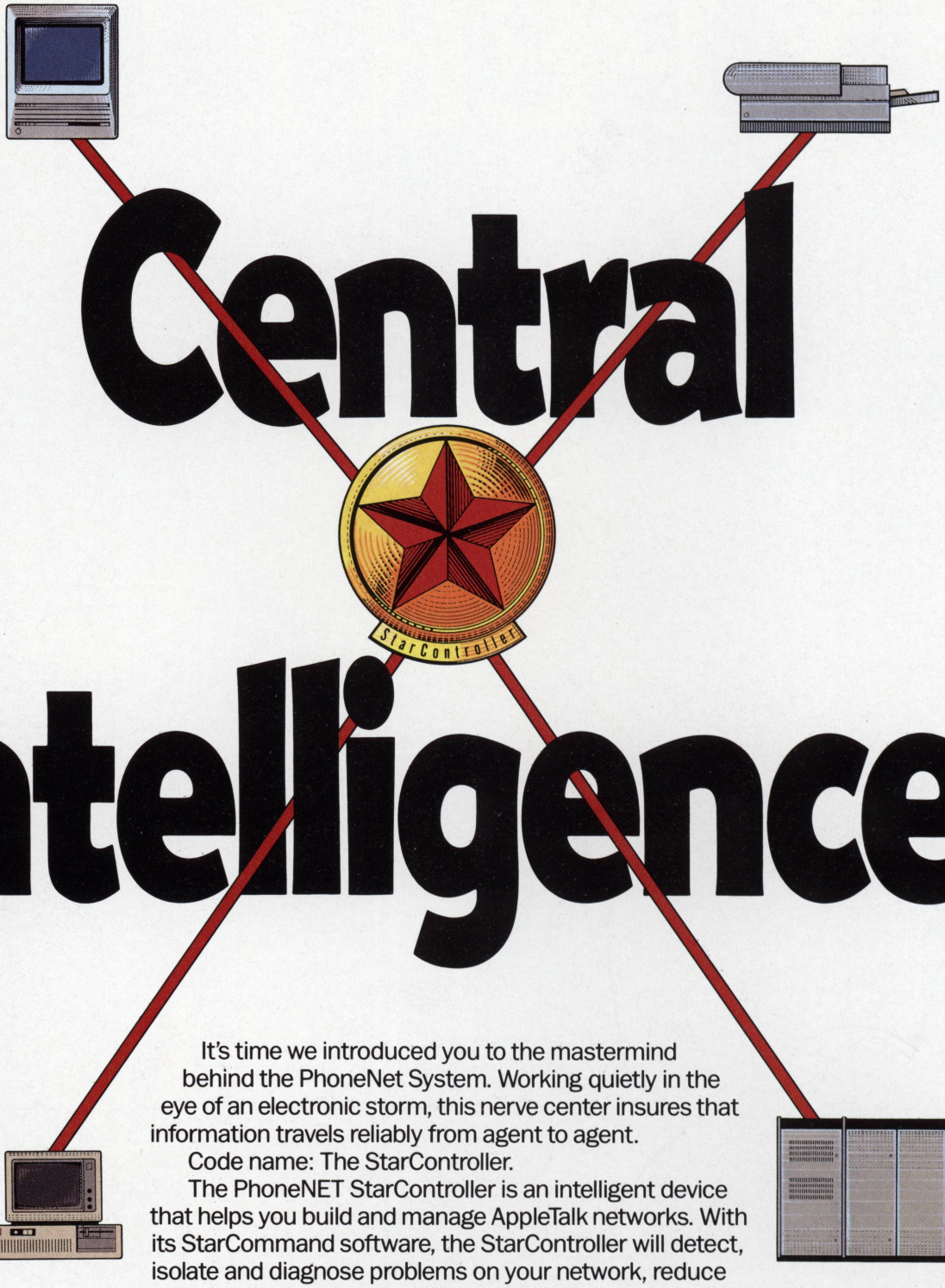
Exploring Hyper*SQL From ORACLE

HyperCard on A Network

Searching Stacks Using Keyword in Context



01
0 74470 74465 8



Central intelligence.

It's time we introduced you to the mastermind behind the PhoneNet System. Working quietly in the eye of an electronic storm, this nerve center insures that information travels reliably from agent to agent.

Code name: The StarController.

The PhoneNET StarController is an intelligent device that helps you build and manage AppleTalk networks. With its StarCommand software, the StarController will detect, isolate and diagnose problems on your network, reduce error rates, and monitor network activity.

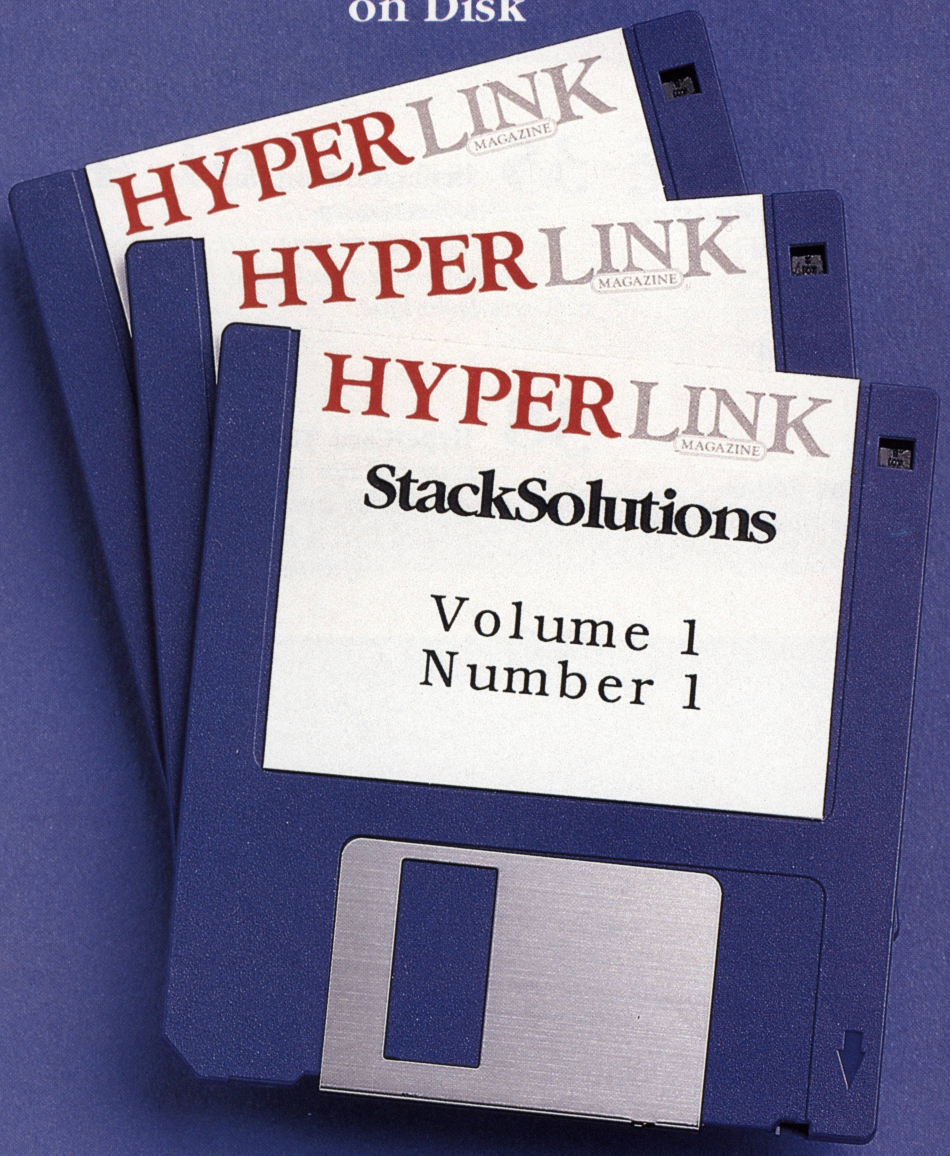
So, should you accept the mission to build a better network, make it a success with the StarController. Your Farallon dealer has a complete dossier. Call (415) 849-2331, for the dealer nearest you.

Farallon™

2150 Kittredge Street, Berkeley, CA 94704

HyperLink Magazine Presents StackSolutions

on Disk



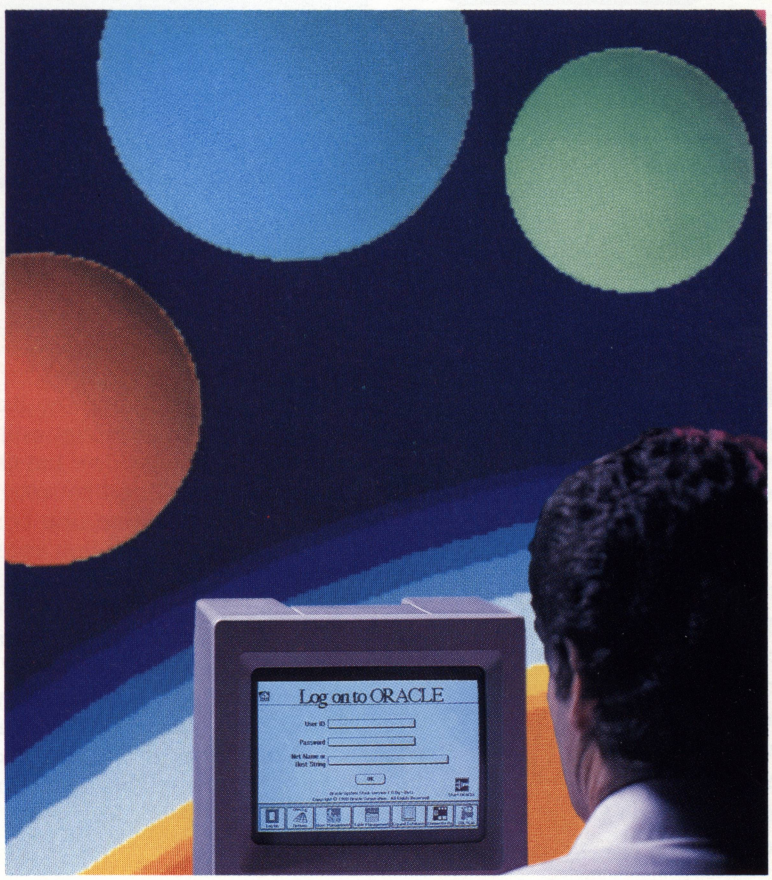
We try to put every bit that we can into the pages of **HyperLink**, but we still find portions of stacks, such as sound and some graphics, impossible to print. In addition, there are cases when we don't have room to print large stacks even though we describe them within the magazine. Our solution is to put everything covered within an issue of **HyperLink** on a **StackSolutions** disk. Every issue of **HyperLink** has a companion **StackSolutions** disk. You may order an individual **StackSolutions** disk or a combination **HyperLink Magazine/StackSolutions** subscription using the order form inside.

HYPERLINK MAGAZINE

COLUMNS

FEATURES

REVIEWS

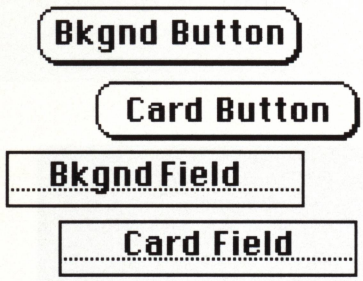


Putting HyperLink Buttons, Fields, And Stacks into Your Mac

How to Use HyperLink

The articles in *HyperLink* are designed to be the best compromise between the use of page space and the graphic representation of what you see on your Mac's screen. In addition, we have tried to make it very easy for you to enter *HyperCard* objects (buttons, fields, etc.) without errors.

For instance, in a button or field description the icon label clearly shows whether the object is in a background or a card layer:



HyperCard objects have a description and possible HyperTalk script associated with them. Here's the standard description format:

Background Button: Home
From Background: Customers
AutoHilite: false
ShowName: false
Visible: true
Icon: 1011
Rectangle: 465,297,502,329
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES
See Listing 1

The description contains all the information to create the object in *HyperCard* on your Macintosh. You will note that the last attribute in the foregoing button description indicates the existence of a script. The object's associated script is listed in a window just like the one on your Mac. The "elevator shaft" and the "white elevator car" indicate the position in the script that is showing in the window:

Listing 1 - Script for background button "Home"

```
put 0 into result
repeat until counter = 0
  get last char of mask
  put it into temp1
  delete last char of ¬
  mask
  get last char of CC
  put it into temp2
  delete last char of CC
```

As you can see, the scripts are visually easy to check as they are entered because they look the same as what appears on your Mac's screen.

Notice the hook character (¬) at the end of line five in the listing (obtained within the script editor window by holding down the Option key and pressing Return). It indicates that the current line continues on the next line. If you enter the line as shown it will work properly. If you find there is enough room for the entire statement to fit on one line when using the *HyperCard* script editor, this character may be omitted. In fact, these characters should only appear at the end of a physical line, otherwise a syntax error will result.

Making it Easier

HyperLink Magazine's staff has created a special *HyperCard* stack called *StackFramer*™, (available on the Volume 1, No. 1 *StackSolutions* disk). This stack makes it easy to enter stacks from our pages. *StackFramer* asks you questions about the object you are entering and leads you through the process of building screen displays identical to those in the magazine.

By visually verifying a button's or field's description character-by-character, line-by-line on the screen with the description on the page, you know it is error free.

StackFramer incorporates a special script entry window for you to enter scripts. This script entry window allows full use of the Edit menu tools unlike the actual HyperTalk script editing window.

After you have given *StackFramer* all of the object definitions (stack, backgrounds, cards, buttons, and fields) and object scripts, it generates the stack for you. You may generate as many copies of the stack as you desire.

If you save the *StackFramer* copy containing the template of a stack, you can later make changes to the template's object descriptions or scripts and generate new stacks that include these changes.

The Easiest Way

If you just don't have the time or inclination to hand enter all of the *HyperCard* objects and stacks, consider purchase of the *HyperLink StackSolutions*™ disk for that magazine issue. Everything is there including special sounds and all graphics.

HYPERLINK MAGAZINE

Jan/Feb 1989 • Volume 2 No. 1

Publisher
David G. Brader

Editor
Roger Wood

Contributing Editors
James Paul
Steve Roti
Rhett Savage
Dan Shafer

Copy Editing
Joan Donaldson

Director of Design
Marv Boggs

Production Artist
Ken Petersen

Director of R & D
William K. Balthrop

Controller
Mark Andersen

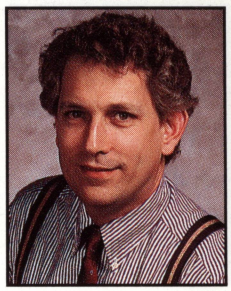
Public Relations
Ken Jackson

Advertising Sales
Carole Eversole
(503) 484-5157

HyperLink Magazine is published 6 times a year by Publishers Guild, Inc., P.O. Box 7723, Eugene, OR 97401. Telephone (503) 484-5157.

Editorial material should be submitted to HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401. Submissions will not be returned.

Publisher's Card



A year has passed since we started publishing *HyperLink Magazine*. It was at the 1988 San Francisco MacWorld Expo that we passed around a sixteen page sample of *HyperLink*. That precursor to our premier issue carried my first "Publisher's Card.{ In it I stated that "*HyperCard* is just the Model T of its genre of software." I penned that thought in November of 1987. Only a year later Steve Jobs introduced his NeXT computer with new software called *NeXTStep*. Is *NeXTStep* the heir apparent to *HyperCard?*

IBM has taken the next step. They have spent millions in the last few months on third party software companies. The hopeful result, a human/computer interface superior to *HyperCard* that can run on Big Blue's PC's. Interfaces that act as gateways to networks of computers from different manufacturers.

It is going to be interesting to observe the evalution of *HyperCard*, *NeXTStep*, Microsoft's *Presentation Manager*, and Metaphor Computer Systems graphic presentation interface (for the IBM OS/2). *HyperLink Magazine* will track these and other interfaces being developed apart from the direct influence of IBM and Apple.

Now that Macintosh has been accepted in big business, corporate users are interested (as are higher education users) in accessing the other worlds of data that exist in their mainframes via graphic-style interfaces. Some of these users will wait until IBM and Metaphor deliver the new graphic interface to IBM's Relational DataBase Management Systems (RDBMS). Others may try to work with interim highbred interfaces on their PC's. Macintosh users may buy *ORACLE for Macintosh* (with its *HyperCard* interface and RDBMS) and start viewing data in new and better ways now. To aid those users, Dan Shafer has another in his series of articles on *ORACLE for Macintosh* in this issue.

The *ORACLE for Macintosh* package allows a *HyperCard* user to access databases almost anywhere, even in the world of Big Blue. That means *ORACLE* databases are now accessible from Apple's LocalTalk networks via *HyperCard*. That's right, *HyperCard* on a network. *HyperCard* can be used over a network in two different ways. *HyperCard* instantaneous communication between two Macintoshes is possible via a new *HyperCard* enhancement package from Gava, Inc, called *HyperCom*. Indirect communication between *HyperCard* machines is also possible via a network file server (like *AppleShare*).

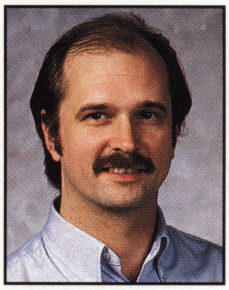
Now you can build a stack that searches bodies of text for a keyword and displays that keyword in the context of each place that it is found. Then, of course, this clever stack allows you to select the specific location of the keyword that you wish to examine by its context. I think you will find this stack to be another valuable tool to add to your growing *HyperLink* StackSolutions.

We hope you enjoy this issue and stay with us as we explore other worlds of data through our *HyperCard* gateway. And, in deed, as we explore other links between these worlds. Remember that *HyperLink Magazine* is

"The One Essential *HyperCard* Resource."

David G. Brader

Letters to the Editor



Roger Wood
Editor

Networking Anyone?

I am on a leave of absence from my school district so that I can pursue other avenues of employment in the computer industry that are not open to me in education. As part of my contract with my school district I still have certain tasks that I must fulfill, including researching the existing supply of Macintosh educational software and especially those programs that would work on the AppleShare Network. Both of the high schools in the district have 25 Mac SE's that are on an AppleShare Network.

I was wondering if *HyperCard* programs can be run on AppleShare Network? Are there *HyperCard* programs that support multi-users? I would certainly appreciate hearing about sources.

By the way your program "Class Stacks" is superb. Between jobs I am trying to fine tune it with *Reports* (from TENPointO) as you suggested so that it is a complete record keeping program for most secondary teachers.

I look forward to any information you may have on this.

Walter R. Schillinger
Oak Park, IL 60302

Glad to oblige, Walter. Most all of your questions are addressed in this issue's "Networking with HyperCard—First Steps," by our publisher Dave Brader. We have instituted sharing data in a number of stacks on our AppleShare Network, and from what we've seen HyperCard *1.2.1 works great. But the article tells much more.*

Changing Tunes?

I really enjoy *HyperLink* and look forward to each new issue. The article "How to Use *HyperLink*" in the first issue says in some articles I will find full-size artwork for scanning and pasting into *HyperCard* stacks. In "How to Use *HyperLink*" in subsequent issues this paragraph has been replaced with an account of the option-return character and none of the promised artwork has appeared. Have you changed your tune?

Speaking of changing your tune and the option-return character, called a soft return (Danny Goodman in The Complete *HyperCard* Handbook), here is a useful tip. the soft return normally only works in command lines, but with a little punctuation it can be made to work in character strings as well. This is handy in quite a few scripting situations. Take a string of notes that may follow a play command. More than a few bars of music will run off the right side of the scripting window, and, although the music will still play, the script will be ungainly and editing the line a headache. Here's a simple example (see Listing 1) of how to change your tune to make it both easier to read and edit.

This tune is from Dan Shafer's HyperTalk Programming. Thanks for your fine publication.

Martin Lewis
Huntington Beach, CA 92646

Thanks for the tip Martin. Yes, we did change our tune on full-size graphics, and you are the first person to have mentioned it. Virtually all of the graphics in our stacks thus far can be easily produced using HyperCard's *painting tools. We feel that scanners are not plentiful enough at this time to warrant the extra space. If you, our readers, would really like this feature we are definitely ready to consider it. What do you think, readers?*

Listing 1

```
Play "Harpsichord" tempo 180 "bq b g g be b" && ¬
"bq d5h a4q a f# f# ae a aq c5h b4q b g g be b " && ¬
"bq d5h a4q be c5e b4q a gh gq"
```

Bugs in "XMAIL"

I have been playing with *HyperCard* for about nine months now, and have gotten good at doing what I want and need to do with it. I recently graduated with a B.S. in Aeronautical Science so computers will most likely only be a hobby for me. I tore my hair out trying to animate simple pictures in *HyperCard*. Slowly with a lot of patience, I figured it out; it all started getting easier. Then I found *HyperLink*. It helped, and gave me all sorts of new ideas. I loved it! I have every issue. I use it. But I became unhappy when I found scripting errors that would drive me crazy trying to figure them out. I had had enough when I did Joan Donaldson's "Short Stacks" in Vol. 1, #3.

I liked the idea she presented. Then I tried it. It didn't work. I finally figured out what was wrong with the script. The "Print Card" button script needed the word background on front of the button for every hide and show command. How annoying! But the very worst part of all of this is the Short Stacks title—"In Praise of *HyperCard's* easy Access to New Users!" A new user would not know what was wrong because he/she typed it in the computer the same way it was in the magazine.

Thanks for letting me get this off my chest; I feel better already!

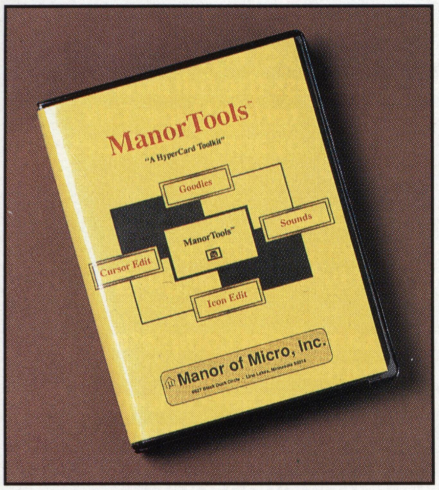
Allen M. Kotarba
Daytona Beach, FL 32018

Right you are, Allen. You're not the only one to have noticed this mistake, and we're sorry that this poorly edited piece of script crept into our magazine. The fix as Allen has stated is simply to place the word `background` *(you could use* `bkgnd` *or* `bg`*) in front of the word* `button` *in every instance in the* `mouseUp` *handler in the right column on page 54 of the September/October issue (Vol. 1, #3). We regret any inconvenience this has caused our readers, especially you beginners.*

StackStarter Elements Removed from *101 Scripts and Buttons*

Just when "share and share alike" seemed to be the credo of the *HyperCard* world, a legal wrangle has surfaced. Robertson Reed Smith, author of *StackStarter* (see *HyperLink*,Vol. 1, #3) and Craig Ragland, author of *101 Scripts and Buttons for HyperCard* (see *HyperLink*,Vol. 1, #4) have a disagreement over the use of parts of *StackStarter* in *101 Scripts and Buttons*. As a result, all of the *StackStarter* parts (a click sound, am inch worm icon, many knobs, and buttons—45 elements according to Robertson) must be removed from *101 Scripts and Buttons*. If you have purchased *101 Scripts and Buttons*, be sure to register. An upgrade is coming, and it will, according to Craig, have new features in place of the removed *StackStarter* elements. By the way, *StackStarter* is still available and improving. Robertson told us that if anyone had difficulty getting *StackStarter* to mail him a SASE and a blank disk and he would send the stack.

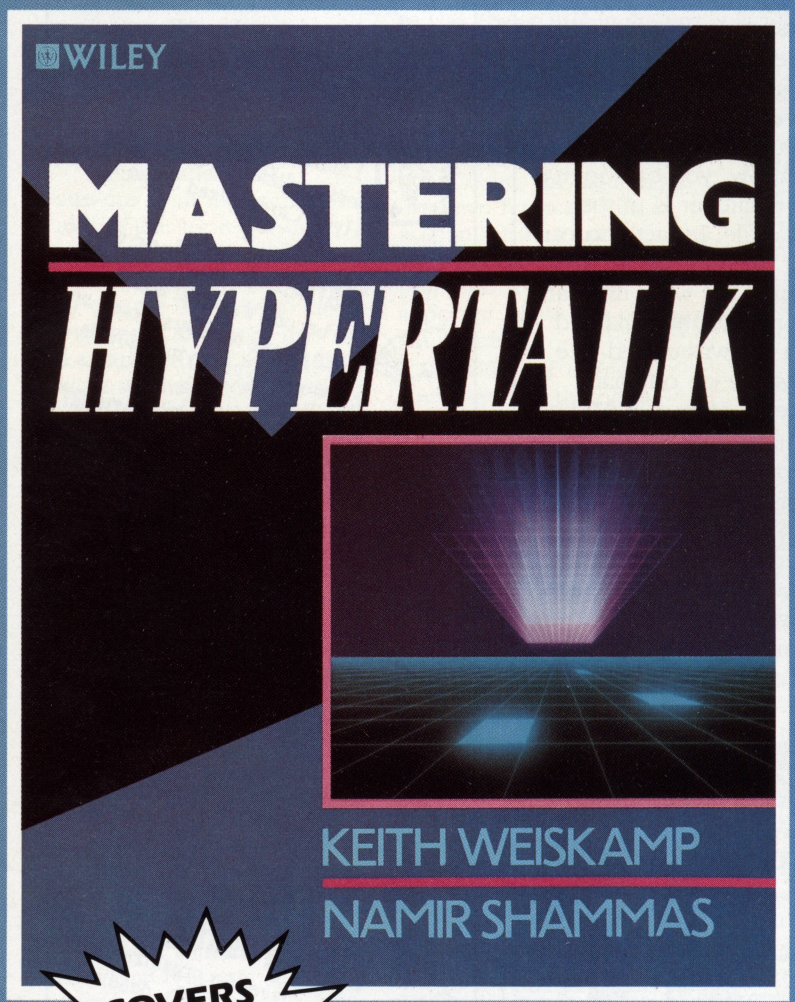
Oops! *Manor Tools* Can Too Transfer Icons & Cursors!



In last issue's "Reviews of HyperCard Products," we stated that the icon and cursor editors in *Manor Tools* from Manor of Micro were "quite serviceable Fat-Bit editors for working on these two important *HyperCard* resource types." This was correct. We then said, "The only real limitation is that you can't load a cursor or icon from one file and save it to another." This was *not* correct—you *can* move icons and cursors. Whenever you have something in the editor window, you can open a file and save whatever is in the window to that file. In fact you can save it to any file, not just a *HyperCard* stack. We were unaware that the editor window remained active when a file was closed. We regret this mistake on our part, and apologize to Manor of Micro.

As was indicated in the review, *Manor Tools* is a fine, easy-to-use product and an excellent value at $59.95. The number of sound effects and its editing capabilities, coupled with some other great "goodies" make it a good all-around utility.

maximum input = maximum output



COVERS VERSION 1.2!

To get the most out of HyperCard, you need to harness the full power of HyperTalk. Mastering HyperTalk gives you all the tips and techniques you'll need.

Major features include:

- A detailed appendix that covers the complete HyperTalk language version 1.2.
- Numerous examples of complete HyperTalk scripts.
- Discussions of important concepts such as object-oriented programming, hypertext, and techniques for developing scripting tools.
- Practical programming topics and algorithms include sorting, searching, graphics programming, and list and string processing.
- Stack applications that you can use or modify to suit your needs.
- Hands-on techniques for adding extensions to HyperTalk.

Get your copy today and get all the power of HyperTalk! $24.95

ABOUT THE AUTHORS: Keith Weiskamp is co-founder and editor of PC AI: The Artificial Intelligence Journal of Personal Computing and the author of Artificial Intelligence Progamming with Turbo Prolog. Namir Shammas, editor of Turbo Tech Report, has co-authored several computer books and written articles for BYTE, Journal of Pascal, Turbo Technix and other leading magazines in the field.

At bookstores or use this coupon to order today.

WILEY
605 Third Avenue
New York, NY 10158-0012

For fastest service, call
1-800-526-5368

ORDER NOW!

JOHN WILEY & SONS, INC. Attn: M. Schustack
605 Third Avenue, New York, NY 10158-0012

Please send me _____ copy(ies) of MASTERING HYPERTALK (1-61593-5) @ $24.95 per copy plus applicable sales tax.

☐ Payment enclosed, Wiley pays postage/handling.
☐ Bill Me ☐ Bill my firm/institution
Bill my ☐ VISA ☐ MasterCard ☐ American Express

Acct. # _____ Exp. _____

Signature _____

Name _____

Company _____

Address _____

City/State/Zip _____

Price subject to change and higher in Canada.

The Nuts And Bolts of A New Macintosh Product

Using ORACLE's Hyper*SQL

by Dan Shafer

Editor's Note: This is the second in a series of articles by HyperTalk columnist Dan Shafer on the use of *HyperCard* as a front end to database systems.

Hyper*SQL, the core of the *ORACLE for Macintosh* product's *HyperCard* interface, extends Structured Query Language (SQL) to accommodate, and take advantage of, the unique power in *HyperCard*. By sing Hyper*SQL, you can design *HyperCard* "windows" into databases residing on any hardware platform (including the Macintosh) on which they happen to be stored.

In the last issue, we discussed the *ORACLE* product and its features. This article explores the technical "guts" of the program. Its purpose is to enable scripters to understand some of the scope and power of Hyper*SQL so that they can determine if they can make use of it in their own projects. This article is not intended to be documentation of Hyper*SQL. It is an overview—neither exhaustive nor penetratingly deep. If after you finish reading this article, you conclude that Hyper*SQL may be useful, you should purchase the developer version of *ORACLE for Macintosh* which is available for less than $200.

The XCMD

Hyper*SQL commands are entered into *HyperCard* stacks by means of the one XCMD incorporated into the product. The XCMD is called `execsql` (for EXECute SQL). It takes a single argument: a string enclosed in quotation marks. It is the composition of this string that constitutes most of the programming demand Hyper*SQL.

The entire SQL command set consisting of the 15 verbs shown in the Figure 1, as well as a few system-level commands and a number of parameters, can be executed by a HyperTalk `execsql` statement. For example, to extract from a database table called "Inventory" the part number, name, and quantity on hand of all the parts used to build a gizmo, you would use this `execsql` statement:

Hyper*SQL Verbs

ALTER	DROP	RENAME
AUDIT	GRANT	REVOKE
COMMENT	INSERT	SELECT
CREATE	LOCK TABLE	UPDATE
DELETE	NOAUDIT	VALIDATE

Figure 1

```
execsql "SELECT PART_NO, NAME," && ¬
"QOH FROM INVENTORY" && ¬
"WHERE USED_BY = GIZMO"
```

When it encounters this command, the `execsql` XCMD assembles the string inside the quotation marks and passes it as a SQL query to the database where the table called "Inventory" is stored.

Using *HyperCard* Containers

Much of the power of the Hyper*SQL design lies in its ability to use *HyperCard* containers as the source of parameters for the SQL command and as destinations for data retrieved using SQL. A *HyperCard* container can be a card, a background field, or a variable. The name of such a container can simply be embedded directly in an `execsql` command line by enclosing it with colons. Any valid *HyperCard* identifier for a container can be used. For example, the inventory-retrieval SQL command in the previous section can be altered to select the name of the device in which the parts are used by re-coding it as:

```
execsql "SELECT PART_NO, NAME," && ¬
"QOH FROM INVENTORY" && ¬
"WHERE USED_BY = :card field 13:"
```

This would permit the user to enter into card field 13 the name of the device whose parts he wants to examine, then to push the button to which the `execsql`

Dan Shafer is the author of HyperLink's *regular "Shafer On Scripting" column. He is also President of Strategy Consulting, the first certified* ORACLE *Mac consultant, and the author of the soon-to-be-published book* Using ORACLE with HyperCard *(Hayden Books). He has been working with Oracle Corp. on the* Hyper*SQL *and* ORACLE for Macintosh *products for several months prior to their announcement.*

statement was attached, and to retrieve information about that device. The real beauty of this approach is that it permits programmers to design often complex SQL commands to retrieve data from a table in such a way that the SQL only needs to be written once. When the parameters of the report change, the manager who uses the database need not go back to the programmer to request a SQL modification; he simply supplies the new parameters in *HyperCard* field. Alternatively, the stack could be designed to use *HyperCard* Ask and Answer dialog boxes for data input.

Listing 1 shows a small `mouseUp` handler that could be used to first ask the user to indicate the device whose parts list he wants to review, and then to extract the data from the table. (We assume the table is open and that the proper housekeeping has already been done so that database access is straightforward.)

Placing the Data

Information retrieved from an *ORACLE* database table can be placed in a prototype card, in named fields, or in named variables.

Prototype cards can be created in one of two ways. If you embed a simple `SELECT` query in an `execsql` statement, Hyper*SQL creates a prototype card for you. It then automatically makes a copy of this prototype card and fills in its fields for each row of the database table retrieved by the `SELECT` command. This means that if you are designing a *HyperCard* interface to an *ORACLE* database where the appearance of the information is secondary or unimportant, you need not even create a card to hold the data. Hyper*SQL handles the assignment for you.

If, for example, you ask Hyper*SQL to execute the following query, you can expect it to create a card with fields like those shown in Figure 2 with no further effort on your part. Hyper*SQL not only creates a field for each column you identify in the `execsql` statement, it also labels them.

```
execsql "SELECT ENAME, SAL
FROM EMP"
```

Listing 1

```
on mouseUp
  ask "What device do you want to look at?"
  if it is empty then exit mouseUp
  put it into card field 13
  execsql "SELECT PART_NO, NAME, QOH FROM INVENTORY"&& ¬
 "WHERE USED_BY = :card field 13:"
end mouseUp
```

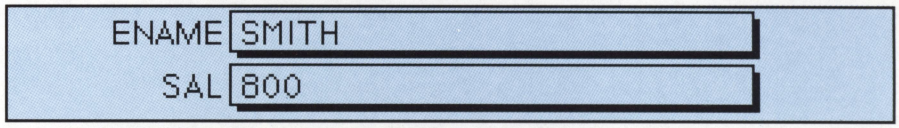


Figure 2

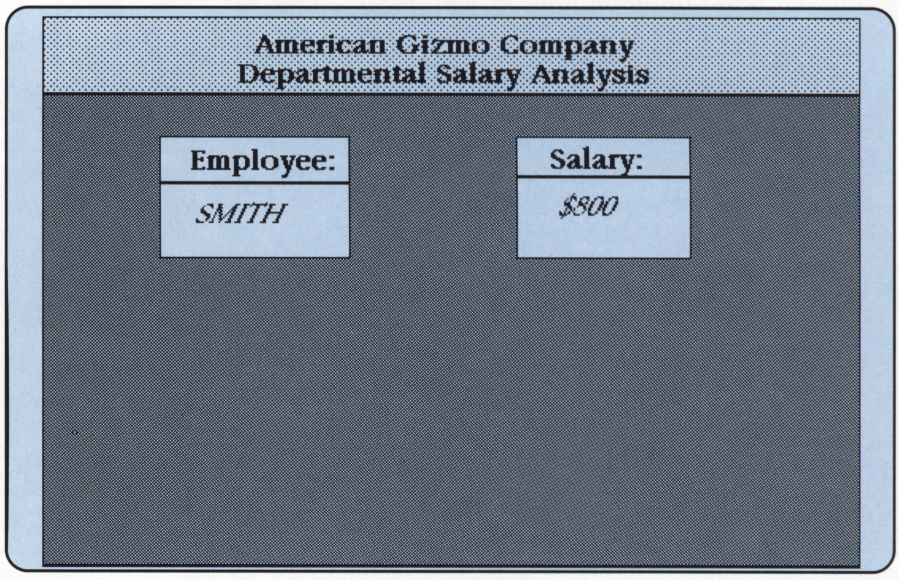


Figure 3

Although this is obviously quite powerful because of the way it streamlines the *HyperCard* design process, it does not take full advantage of *HyperCard's* ability to make data look interesting. To do this, you can create your own prototype card and force Hyper*SQL to use it rather than creating its own by including the `USING` clause in your `execsql` statement. For example, the following query will use the prototype card shown in Figure 3 to produce a far more pleasing *HyperCard* display.

```
execsql "SELECT ENAME," && ¬
"SAL FROM EMP USING" && ¬
":protocard 1:"
```

Obviously, you could design and build a number of enhancements to the card design this way. The only limit is what *HyperCard* permits you to do.

When you design a prototype card, you must supply information for each field telling Hyper*SQL

- the *name* of the column from the database table to be stored in that field
- the *type* of data it contains (SQL; unlike HyperCard, uses data with defined types),
- the *size* of the value, and
- the *width* of the data to be displayed.

Designing a prototype card has another interesting advantage. Even if you use the SQL command `SELECT *`, which means you want to select all of the columns in the table for each row that meets the criteria, Hyper*SQL only retrieves fields named on the prototype card in the `USING` statement. This means that the following command has exactly the same effect as the longer one shown above:

```
execsql "SELECT *" && ¬
"FROM EMP USING" && ¬
":protocard 1:"
```

Listing 2

```
-- Partial listing only.
-- Not a fully working script.
-- Assumes a series of card-level radio buttons
-- named "Report 1", "Report 2," etc., which the
-- user highlights before pressing the button to
-- which this script is attached.  Prototype cards
-- have names corresponding to the button names.
on mouseUp
  repeat with counter = 1 to 6
    if the hilite of card button counter is true then
      put the short name of card button counter ¬
      into rName
      exit mouseUp
    end if
  end repeat
  put "card" && rName into cardToUse
  execsql "SELECT * FROM EMP USING :cardToUse:"
end mouseUp
```

Listing 3

```
-- Partial script, not intended to work as presented
on mouseUp
  execsql "SELECT ENAME, SAL INTO :bg fld Name:," && ¬
  ":salNow:"
  put salNow * 1.115 into newSal
  put salNow into bg fld Salary1
  put newSal into bg fld Salary2
end mouseUp
```

One place this feature can be used to great advantage is in situations where you have several different formats for data, each requiring different sets of columns to be retrieved from the same table. Rather than write a separate `execsql` statement for each retrieval, you could just determine the format to be used and employ one `execsql` statement referencing the appropriate prototype card. Listing 2 is part of a script that could be used to accomplish this. The prototype cards can each have a different format calling for different columns to be retrieved, and the `execsql` statement shown in Listing 2 will work for all of them.

In addition to placing the data retrieved by a `SELECT` command into a prototype card created by Hyper*SQL or by you, you can also send the retrieved data to named fields or containers. You do this using the Hyper*SQL `INTO` clause. For example, assume that you want to show the user what employees' new salaries will be if they are given a raise of 11.5 percent. The partial script in Listing 3 shows how to do it.

Retrieving Records

None of the `execsql` statements we've seen so far will actually retrieve any data from an *ORACLE* database table. They establish the framework for such retrieval, but the actual retrieval of data is performed by the statement:

```
execsql "GET NEXT ROW"
```

This is because SQL permits you to move forward through a database but not to backtrack through it without programming effort.

The `GET NEXT ROW` command is similar to a HyperTalk repeat statement, though it is sometimes used inside such a repeat loop. Each time the `execsql` statement carries out the `GET NEXT ROW` instruction it updates the *HyperCard* special container `the result`. In normal operation, `the result` will have one of two values when it is tested after employing an `execsql` statement:

- 0, if the row has been read and no error occurred
- 24, if the last row has been retrieved

Listing 4 shows how the GET NEXT ROW operation is carried out in a HyperTalk script. Listing 4 is identical to Listing 2 except for the three lines in the box near the end of the handler.

Figure 4 depicts the process by which a HyperTalk script using execsql statements, and a prototype card, can be used to create a new card for each row read from the database table.

But, if the user is going to retrieve a very large number of rows from one or more tables and does not wish to keep them in *HyperCard* for manipulation but only wishes to examine them on the screen, your HyperTalk script will want to buffer the records being read in and reuse the same prototype card copy repeatedly. Listing 5 is a partial listing that demonstrates the strategy for accomplishing this task.

In network operations, Hyper*SQL automatically buffers 20 rows of data into your Mac's memory. (You can adjust this number up or down depending on performance and other considerations.) This limits the impact on the network by reducing the number of network accesses your Mac makes during database management operations.

Other SQL Operations

Besides retrieving data, you can use the execsql XCMD to change information in an *ORACLE* database and to manage various system-level tasks such as:

- opening and closing databases
- managing database changes and their cancellation
- setting parameters
- dealing with errors

(In the next instalment, we will focus on commonly used SQL commands and their use in Hyper*SQL applications.)

Listing 4

```
-- Partial listing only.
-- Not a fully working script.
-- Assumes a series of card-level radio buttons
-- named "Report 1", "Report 2," etc., which the
-- user highlights before pressing the button to
-- which this script is attached.  Prototype cards
-- have names corresponding to the button names.
on mouseUp
  repeat with counter = 1 to 6
    if the hilite of card button counter is true then
      put the short name of card button counter ¬
      into rName
      exit mouseUp
    end if
  end repeat
  put "card" && rName into cardToUse
  execsql "SELECT * FROM EMP USING :cardToUse:"
  repeat while the result = 0
    execsql "GET NEXT ROW"
  end repeat
end mouseUp
```

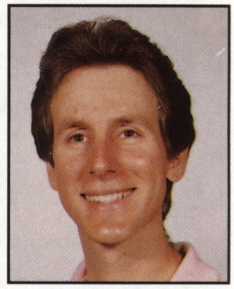
Listing 5

```
-- Partial listing only.
-- Not intended to run as presented.
-- Adapted from ORACLE System DBA Stack script
-- by Mike Roberts
-- Background field "RowNumber" is used to display to
-- user the number of the currently displayed record.
-- This script is executed each time a button named
-- "Next Row" is pressed.
domenu "delete card"
get word 2 of background field "RowNumber"
add 1 to it
put it into word 2 of background field "RowNumber"
execsql "next row"
```

Statement	HyperCard Actions	ORACLE Actions
SELECT * FROM EMP USING :card1:	None	None
SELECT * FROM EMP	None	Creates prototype card with one field per database table row
REPEAT WHILE THE RESULT = 0 execsql "GET NEXT ROW"	Creates new card by copying prototype card and preparing to receive new data	Retrieves next row of database table and places results into appropriate fields on prototype card

Figure 4

Expanding HyperCard Searches by...



Searching Stacks with Keyword in Context

by Steve Roti

Locating a particular bit of information can be difficult whether it is stored on a computer or not. Sometimes the sheer quantity of data to be searched is overwhelming; other times you just forget into which file you put something. Information stored in a *HyperCard* stack is reasonably accessible because of *HyperCard's* fast text searching abilities, but even the venerable `find` command has its limitations. Here's a stack employing a technique known as "Keyword in Context" searching.

The Fight for Information Access

A major battle in the personal computer revolution has been fought over easy access to information. This battle has been won. Now we have complete reference works via CD-ROM disks, world-wide electronic mail, and on-line information services of all kinds. In fact, we have access to so much information that it is easy to get lost in the midst of it all. We find ourselves wandering through aisle after aisle of the information supermarket without being able to locate what we want. To visualize this kind of problem consider the encyclopedia (the paper version, not the electronic one). There are two ways we are able to find information quickly in an encyclopedia:

- Topics are arranged in alphabetical order
- Keywords within topics are alphabetically cross-referenced in the index

In the index is usually the first place we look when trying to find something in an encyclopedia; without it we would be unable to find anything other than the main alphabetized topics. The problem with much of electronic information is that it is not indexed, or at least not as well as we would like it to be. The "Help" stack for *HyperCard* is an example of a large body of information that is not comprehensively indexed.

Steve Roti is a database consultant and writer. He is the owner of Olympic Software, P.O. Box 8989, Portland, OR, 97207; MCIMAIL SROTI. Steve has written for PC Magazine, Data Based Advisor, and other leading magazines.

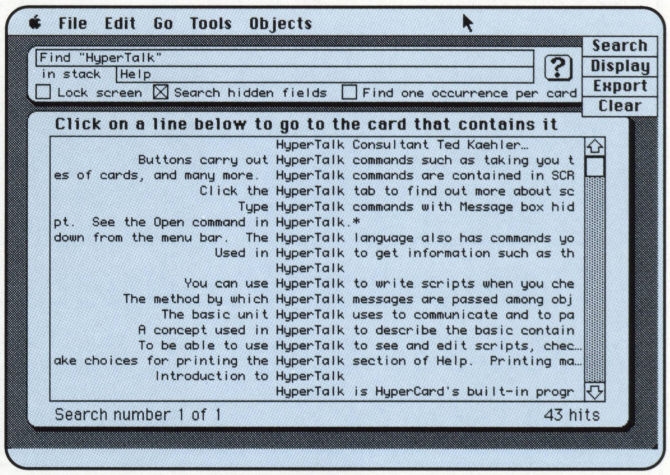


Figure 1

This card shows the design of the "Keyword in Context" stack's main card and background. All objects are placed in the background so just copying this card, or creating a new stack with this background, copies all scripts and buttons.

Apple realized the need for an index and created the "Help Index" stack, but there are many topics and keywords missing from this index.

Lack of indexing is not the only problem with electronic information. Unlike printed words, data stored on a computer disk is dynamic. It can change as old information is modified, or deleted, and new information is added. This makes it impractical to use a static, unchanging index with *HyperCard* data. A better method would be to create an index at the time of the search.

KWIC

The solution to both of these problems is Keyword in Context (KWIC) searching. I first encountered KWIC in graduate school when my advisor was using it to index medical reference texts that had been put in text files. His approach was to print on paper an exhaustive index of all keywords in the references. The classic format of a KWIC index displays one instance of a keyword per line, with the alphabetized keywords centered on the line and contextual information to the left and right. Users would find the keyword references that interested them based on the context in which

Listing 1 - Building A List Found Occurrences within A Stack

```
put empty into firstFound
put empty into searchList
repeat
  do background field "findCommand"
  if the result is "not found" then exit repeat
  -- test whether a full circle has been done
  if the foundChunk && "of" && the name ¬
  of this card is firstFound then exit repeat
  -- set firstFound the first time through
  if firstFound is empty then ¬
  put the foundChunk && "of" && the ¬
  name of this card into firstFound
  put the foundLine after searchList
end repeat
```

Listing 2 - This Modification of Listing 1 Provides for Searching More Than One Stack.

```
put empty into searchList
repeat with stackNumber = 1 to the number of lines ¬
in background field "stacks"
  put empty into firstFound
  repeat
    -- inner loop same as above
  end repeat
end repeat  -- end of outer loop
```

they occurred, and then use the equivalent of the `find` command to jump to the appropriate place in the text. This is an example of static indexing which worked well because the reference information rarely, if ever, changed.

About The "Keyword in Context" Stack

I have adapted the KWIC concept to *HyperCard* in the "Keyword in Context" stack. The major difference between my approach and the classic approach is that the stack builds small dynamic indexes rather than large static indexes when you are ready to search. In order to make searches acceptably quick, the "Keyword in Context" stack only indexes the keyword(s) in which you are interested. In brief, you enter a `find` command and the stack finds all occurrences of the keyword(s), then presents the results to you in a KWIC window—a scrolling "searchList" field. Figure 1 shows the results of searching the *HyperCard* "Help" stack for the keyword "HyperTalk."

The first field in the upper part of the card is named the "findCommand" field. Any valid *HyperCard* `find` command can be typed into this field, thus giving the user plenty of flexibility to search for characters, a word, a string, or multiple words. During searches the `find` command is run repeatedly, using the `do` command as shown in Listing 1.

If there are no occurrences of the keyword, the `repeat` loop terminates immediately. If there is one or more occurrence, the `repeat` loop continues until the first keyword occurrence is found a second time. This indicates that the `find` command has made a complete circle through the stack. (This code example doesn't show the logic for centering the keyword in the `foundLine`—see Listing 9 for the complete routine.) After the `repeat` loop ends, the `searchList` variable can be displayed in the "searchList" field. Those are the basics of KWIC searching in *HyperCard*: all that is needed is a `find` command and a stack name. But there are some additional options to make this type of searching more powerful.

Advanced Features

The "Keyword in Context" stack can search more than one stack at a time. The second field on the upper part of the card is named the "Stacks" field. For simple searches, you enter a single stack name in this field. If you want to search through multiple stacks, just press the Enter key and the field is expanded into a 15 line scrolling field. You can enter as many stack names as you want, one per line. When you finish entering the stack names, press the Enter key again and the field shrinks back down to one line; however, the "in stacks" label in front of the field indicates that there are now multiple stacks in the list.

The stacks are searched in the order that you enter them in the list and all of the keyword occurrences are shown in the KWIC window at the bottom of the card in the order that they were found. In the HyperTalk code, all we need to do to search multiple stacks is to add an outer `repeat` loop and move the initialization of the `searchList` variable in front of the outer loop (the `firstFound` variable still needs to be initialized for every stack).

Three check boxes underneath the "Stacks" field give you finer control over how the search is done. The "Lock screen" button lets you choose not to see the search as it is being done. ("Lock screen" also makes the search go a little faster.) The "Search hidden fields" button lets you search for keywords in fields that are not visible. By default, the KWIC search does not find keywords in hidden fields. The "Find one occurrence" button lets you choose to find only the first occurrence of the keyword on each card. This option can dramatically speed up the search while still finding all cards that contain the keyword. You, of course, lose some contextual information when a word appears more than once on a card.

After a search has been done, there are quite a few things you can do with the list displayed in the KWIC window. You can click on one of the lines in the window to go to the card that contains that occurrence of the keyword. The keyword will be highlighted on the card. If you option-click on one of the lines in the KWIC window, a pop-up window vertically centered on the line you clicked on shows you the details

(card name, field name, and which characters) of where that occurrence of the keyword was found. Click once on the pop-up window to dismiss it.

The HyperTalk code necessary to center and display the pop-up window uses the `clickV` function to determine the vertical location of the mouse click:

```
set the top of background ¬
field "searchDetails" ¬
to the clickV - 12
show background field ¬
"searchDetails"
```

There are two buttons underneath the "Search" button that make use of the entire KWIC list. The "Display" button flips through the cards containing the keyword, pausing for one second and highlighting the keyword on each one. The card display can be stopped with a mouse click. The "Export" button writes the details for all the keyword occurrences to a text file. The text file can be printed with a word processor.

When you are done with the results of a KWIC search, press the "Clear" button and the card is set up for you to specify a new set of search criteria.

Tricking The "go" Command

Here's a tip on how to coerce the `go` command into helping you find a stack when you are not sure of its name. When *HyperCard* cannot find a stack that it has been told to go to, it puts a standard "Open" dialog box on the screen and asks you "Where is" the stack. (The standard "Open" dialog box is the same one that you see when you choose "Open Stack..." under the File menu.) You can navigate through the folders until you find the stack you want or until you decide to cancel. We can make use of this feature by slipping the `go` command a bogus stack name with the intention of choosing the correct stack name in the resulting dialog box.

```
lock screen
go to ¬
"the stack you want to go to"
if the result is "Cancel"
then
  put "Cancel" into msg
else
  put the short name of ¬
  this stack into myStack
  go back
  -- return to where
  -- we came from
  put myStack into msg
  -- display stack name
end if
```

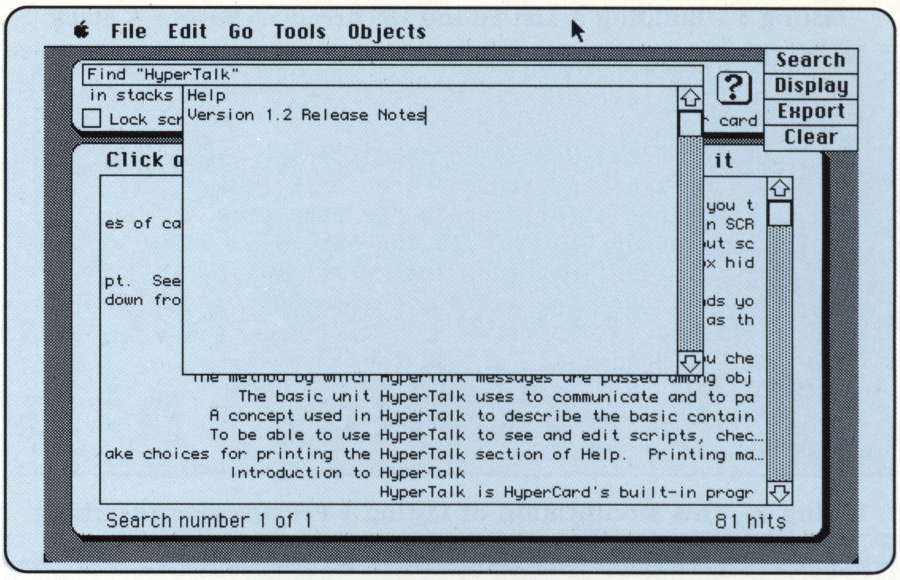


Figure 2

If you wish to enter more than one stack name in the "stacks" field, just click in the field and press the Enter key, and the field expands to a 15 line scrolling field.

Here the `go` command displays the standard "Open" dialog box and asks "Where is the stack you want to go to?" If you choose a stack, its name appears in the Message box. If you click the "Cancel" button, the Message box shows the word "Cancel" instead. Many variations of this trick are possible. The "Keyword In Context" stack uses it when the "Search" button is pressed before a stack name has been entered. After the search, the stack name is automatically put into the "Stacks" field. For a method that employs an XFCN to make compiling the stack list even easier see the sidebar "Selecting Stacks Using The 'fileName' XFCN" on the next page.

Bug in *HyperCard 1.2?*

I discovered a potential bug in *HyperCard 1.2* while developing this stack. I needed to use version 1.2 to build the "Keyword in Context" stack, because it was the first version of *HyperCard* with HyperTalk functions that indicate where a keyword was found in a stack (`the foundChunk` and `the foundLine`).

While I was testing the stack on a large hard disk with many big stacks on it, I discovered that once in a while these functions did not return a value when they should (i.e., when something was found). These missing function values appear to coincide with prolonged disk accesses by *HyperCard*, and they are not repeatable from one search to the next. My workaround in the "Keyword in Context" stack is to put the function values into variables immediately after a successful find; I then test the variables to make sure that they contain something other than `empty`. If one of them is `empty`, that occurrence of the keyword is ignored. If this actually is a bug and not some sort of weirdness on the Mac that I performed my testing on, I hope that it will be fixed in an upcoming release.

Conclusion

I hope this stack helps you in managing your *HyperCard* data. Keep in mind the Keyword in Context searching is most effective on stacks with text paragraphs stored in large fields. It is much less useful on database-type stacks where each type of information has its own pigeonhole. The "Keyword in Context" stack available on the *StackSolutions* disk for this issue contains a "Help" card that is reached by clicking on the "?" button. This

card contains additional instructions about Keyword in Context searching.

I plan to enhance this stack in the future, and I would be interested in hearing your suggestions for improvement. Thanks to Jim Ault and George Champlin of the Portland Macintosh Users Group for suggestions that have been incorporated into the current version.

Stack Name:
Keyword In Context 1.8
Contains Backgrounds:
Keyword Search

Stack Script: Yes
See Listing 3

Background: Keyword Search
Background Script: Yes
See Listing 4

Bkgnd Field

Background Field: searchNumber
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: true
WideMargin: false
AutoTab: false
Rectangle: 35,307,235,324

Selecting Stacks Using The "fileName" XFCN

The ability to select multiple files to be searched provides the "Keyword in Context" stack with power. Having to remember the exact file names for a number of files, however, detracts greatly from the stack's ease of use. There is an XFCN—fileName—that can help simplify the task of building this list. Written by Steve Maller of Apple Computer, this XFCN was originally made available to the general public on large BBS systems in November of 1987. This early version is still available as FILENM.SIT on CompuServe, and an updated version was recently released with the *HyperCard VideoDisc Toolkit* from APDA. [See the overview of this package in *HyperLink Magazine*, Vol. 1, No. 4 for price and availability—Ed.]

The fileName XFCN enables the HyperTalk scripter to employ the standard "Open" Dialog box to prompt the user to select a particular file, and the XFCN returns the full path name of that file. Using fileName in a repeat loop makes it is easy to obtain a whole list of files.

By installing this XFCN in the "Keyword in Context" stack, or in your "Home" Stack, you can use a short script to build your "stacks to be searched" list, and ensure that the names are correct, and that the files really exist where your stack can find them. This installation can be accomplished using *ResEdit* (available from APDA) or *ResCopy* (available from a number of sources, including the *HyperCard VideoDisc Toolkit* mentioned above). We have included a version of the stack called "Keyword in Context w/fileName" on the *StackSolutions* disk for this issue. We simply put the following script in the small "stackPrompt" field (just to the left of the "stacks" field shown in Figure 2)

```
on mouseUp
  repeat forever
    put (the number of lines in field "stacks") ¬
    + 1 into currentLine
    put fileName("STAK") into newName
    if (newName is not empty)
    then put newName & return into line ¬
    currentLine of field "stacks"
    else exit repeat
  end repeat
  repeat with i = the number of lines in me ¬
  down to 1
    if line i of me is empty then delete ¬
    line i of me
  end repeat
  if the number of lines in field  "stacks" < 2
  then put "in stack" into field ¬
  "stackPrompt"
  else put "in stacks" into field "stackPrompt"
  select after line 1 of field "stacks"
end mouseUp
```

—HLM Staff

Style: transparent
TextAlign: left
TextFont: Geneva
LineHeight: 16
TextSize: 12
TextStyle: plain
Field Script: No

Bkgnd Field

Background Field: findCommand
From Background:
Keyword Search
LockText: false
ShowLines: false
Visible: true
WideMargin: false
AutoTab: true
Rectangle: 24,30,415,44
Style: rectangle
TextAlign: left
TextFont: Monaco
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: No

Bkgnd Field

Background Field: hits
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: true
WideMargin: false
AutoTab: false
Rectangle: 385,307,471,324
Style: transparent
TextAlign: right
TextFont: Geneva
LineHeight: 16
TextSize: 12
TextStyle: plain
Field Script: No

Bkgnd Field

Background Field: searchList
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: true
WideMargin: false
AutoTab: false
Rectangle: 35,99,471,306
Style: scrolling
TextAlign: left
TextFont: Monaco

Listing 3 - Stack Script for "Keyword in Context" Stack

```
on openStack
  global oldUserLevel
  if the version < 1.2 then
    answer ¬
     "This stack requires HyperCard 1.2 or later" ¬
      with "OK"
    go home
    exit openStack
  end if
  hide msg
  show menuBar
  put the userLevel into oldUserLevel
  set the userLevel to 5
end openStack

on closeStack
  global oldUserLevel
  set the userLevel to "oldUserLevel
end closeStack

function clickLine
  return trunc((scroll of the target + the clickV ¬
  - item two of the rect of the target) ¬
  div the textHeight of the target) + 1
end clickLine
```

Listing 4 - Script for Background "Keyword Search"

```
on openCard
  hide the message box
  hide field "searchDetails"
  show menuBar
  put "Search number" && the number of this card && ¬
  "of" && the number ¬
  of cards - 1 into field "searchNumber"
  if the number of lines in field "stacks" > 1 then
    put "in stacks" into field "stackPrompt"
  else
    put "in stack" into field "stackPrompt"
  end if
  if field "hits" is empty then
    put "Enter a Find command and stack name, " & ¬
    "then click on Search" into field "instructions"
    put "Find" && quote & quote into field ¬
    "findCommand"
    select after char 6 of field "findCommand"
  else
    put "Click on a line below to go " & ¬
    "to the card that contains it" ¬
    into field "instructions"
  end if
end openCard
```

LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: Yes
See Listing 5

Bkgnd Field

Background Field: instructions
From Background:
Keyword Search
LockText: true

Listing 5 - Script for Background Field "searchList"

```
on mouseUp
  put (the optionKey is down) into showDetails
  put clickLine() into clickedLine
  if line clickedLine of field "searchList" is not ¬
  empty then
    select line clickedLine of field "searchList"
    if showDetails then
   -- Option key was down so show search details
      if last character of line clickedLine of field ¬
      "searchList" is "…" then
        put "(hidden)" into hidden
      else
        put empty into hidden
      end if
      put clickedLine & "." & return & line ¬
      clickedLine of field "searchList" & return & ¬
      line clickedLine of field "searchChunk" ¬
      && hidden & return & line clickedLine of field ¬
        "searchCard" into field "searchDetails"
      set the top of field "searchDetails" to ¬
      the clickV - 12
      show field "searchDetails"
    else
    -- Option key not down so go to keyword occurrence
      put line clickedLine of field "searchChunk" ¬
      into theChunk
      go to line clickedLine of field "searchCard"
      if the result is "No such card" then exit mouseUp
      select theChunk
    end if
  end if
end mouseUp
```

Listing 6 - Script for Background Field "searchCard"

```
on mouseUp
  set lockText of me to false
  click at (item 1 of rect of me + 2),(item 2 of the ¬
  clickLoc)
  click at (item 3 of rect of me - 18),(item 2 of the ¬
  clickLoc) with shiftKey
  set lockText of me to true
  set lockScreen to true
  -- so that the selection stays hilighted
  if the selection is empty then
    set lockScreen to false
    beep
    answer "That is not a card name.  Try again!"
  else
    put background field "searchFor" into searchString
    go to the selection
    do searchString
    set lockScreen to false
    put searchString into the message box
  end if
end mouseUp
```

ShowLines: false
Visible: true
WideMargin: false
AutoTab: false
Rectangle: 35,81,470,98
Style: opaque
TextAlign: left
TextFont: Geneva
LineHeight: 16
TextSize: 12
TextStyle: bold
Field Script: No

Bkgnd Field

Background Field: searchCard
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: false
WideMargin: false
AutoTab: false
Rectangle: 35,99,471,306
Style: scrolling
TextAlign: left
TextFont: Geneva
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: Yes
See Listing 6

Bkgnd Field

Background Field: searchChunk
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: false
WideMargin: false
AutoTab: false
Rectangle: 35,99,471,306
Style: scrolling
TextAlign: left
TextFont: Monaco
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: Yes
See Listing 7

Bkgnd Field

Background Field: searchDetails
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: false
WideMargin: false
AutoTab: false
Rectangle: 2,225,510,277
Style: scrolling
TextAlign: left
TextFont: Monaco
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: Yes

```
on mouseUp
  hide me
end mouseUp
```

Bkgnd Field

Background Field: stacks
From Background:
Keyword Search
LockText: false
ShowLines: false
Visible: true
WideMargin: false
AutoTab: true
Rectangle: 87,43,415,57
Style: rectangle
TextAlign: left
TextFont: Monaco
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: Yes
See Listing 8

Bkgnd Field

Background Field: stackPrompt
From Background:
Keyword Search
LockText: true
ShowLines: false
Visible: true
WideMargin: false
AutoTab: false
Rectangle: 24,43,88,57
Style: transparent
TextAlign: left
TextFont: Monaco
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: No

Bkgnd Button

Background Button: Search
From Background:
Keyword Search
AutoHilite: true
ShowName: true
Visible: true
Icon: none
Rectangle: 452,19,512,36
Style: rectangle
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain

Listing 7 - Script for Background Field "searchChunk"

```
on mouseUp
  put clickLine() into clickedLine
  if line clickedLine of background field ¬
  "searchList" is not empty then
    select line clickedLine of background field ¬
    "searchList"
    put background field "findCommand" into findCommand
    go to line clickedLine of background field ¬
    "searchJump"
    do findCommand
  end if
end mouseUp
```

Listing 8 - Script for Background Field "stacks"

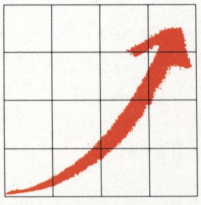
```
on enterInField
  lock screen
  if style of me is "rectangle" then
    set style of me to "scrolling"
    set the rect of me to 87,43,415,225
    put "in stacks" into field "stackPrompt"
    if return is not in me then put return after me
    unlock screen
    select after text of me
  else
    set style of me to "rectangle"
    set the rect of me to 87,43,415,57
    repeat with i = the number of lines in me down to 1
      if line i of me is empty then delete line i of me
    end repeat
    if the number of lines in field "stacks" < 2 then
      put "in stack" into field "stackPrompt"
    end if
    unlock screen
    select after line 1 of me
  end if
end enterInField
```

Listing 9 - Script for Background Button "Search"

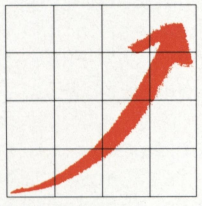
```
on mouseUp
  put "                                " into blankString
  -- put 30 spaces between the quotes above
  put the hilite of bkgnd button ¬
  "Search Hidden Fields" into searchHidden
  put the hilite of bkgnd button ¬
  "Find One Occurrence" into findOne
  put the hilite of bkgnd button "Lock Screen" ¬
  into withLockScreen
  -- validate the find command
  if field "findCommand" is "Find" && quote & quote ¬
  or word 1 of field "findCommand" is not "Find" or ¬
  the number of words in field "findCommand" < 2 then
    beep
    answer "Invalid Find command!  Please try again"
    put empty into field "hits"
    openCard
    exit mouseUp
  else
    put field "findCommand" into findCommand
  end if
  if field "stacks" is empty then
```

Find out what's up at MACWORLD Expo.

Attendance 1985 - 1988

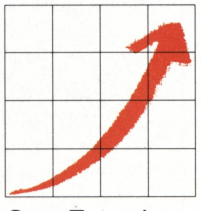


San Francisco
Up 144%

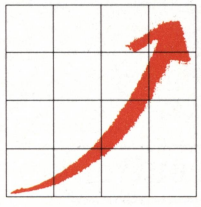


Boston
Up 166%

Expo Booths 1985 - 1988



San Francisco
Up 300%



Boston
Up 280%

You can always count on seeing the newest, the best and the most at MACWORLD Expo. That's why, year after year, the original Macintosh-exclusive computer show keeps getting more and more popular, both for users and for vendors.

Fact is, MACWORLD Expo will:

• **Save you time and money . . .** by literally putting at your fingertips the hardware, software and peripherals that represent the entire state of the art of Macintosh™ computing. *You'll make smarter buying decisions.*

• **Build your knowledge . . .** by enabling you to attend a helpful tutorial that meets your specific need, whether you use your Mac at the office, at school or at home. *You'll learn from the experts.*

• **Develop your skills . . .** by giving you plenty of opportunities to practice what you've learned, using one of the many Macintoshes that will be available to you. *You'll get hands-on experience.*

The number of MACWORLD Expo cities is up too.

Now our nation's capital joins San Francisco and Boston in hosting MACWORLD Expo. You'll see special emphasis on the products, systems and applications representing the specific needs of government agencies and operations. Pick the location that's best for you:

SAN FRANCISCO January 20-22, 1989
Moscone Center • Brooks Hall • Civic Auditorium

WASHINGTON, DC April 26-28, 1989
DC Convention Center

BOSTON August 10-12, 1989
Bayside Expo Center • World Trade Center

The next move is up. Just fill in, detach and return the coupon below to MACWORLD Expo, Box 155, Westwood, MA 02090. We'll mail you the information you need to: get a special reduced rate on your admission, avoid the registration lines and make the most of your time learning what's up.

What's up, MAC?

Please send me details about attending the following MACWORLD Expos:

❑ **San Francisco**, Jan. 20-22, 1989 ❑ **Washington, DC** April 26-28, 1989 ❑ **Boston**, August 10-12, 1989

Name _____ Company _____

Street _____ City _____

State _____ Zip _____ Phone _____ HYP

Sponsored by MACWORLD, the Macintosh™ Magazine. An IDG Communications publication. MACWORLD Expo is an independent trade show not affiliated with Apple Computer, Inc. MAC, MACINTOSH and MACWORLD are trademarks of Apple Computer, Inc.

MACWORLD EXPOSITION®

Button Script: Yes
See Listing 9

Bkgnd Button

Background Button: Display
From Background: Keyword Search
AutoHilite: true
ShowName: true
Visible: true
Icon: none
Rectangle: 452,35,512,52
Style: rectangle
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: Yes
See Listing 10

Bkgnd Button

Background Button: Help
From Background: Keyword Search
AutoHilite: false
ShowName: false
Visible: true
Icon: 25002
Rectangle: 422,33,446,58
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: Yes

```
on mouseUp
  go to card "Help"
end mouseUp
```

Bkgnd Button

Background Button: Export
From Background: Keyword Search
AutoHilite: true
ShowName: true
Visible: true
Icon: none
Rectangle: 452,51,512,68
Style: rectangle
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: Yes
See Listing 11

Listing 9 - Script for Background Button "Search"

```
  -- set up prompt for the stack name
   put "the stack you want to search" into searchStack
else
   put field "stacks" into searchStack
end if
push card
set cursor to busy
put empty into searchList  -- KWIC list
put empty into searchCard  -- pointers to cards
put empty into searchChunk  -- pointers to chunks
put 0 into hits
put 150 into maxHits  -- must not exceed 450
put empty into field "searchList"
put "Searching..." into field "hits"
set the lockRecent to true
set the lockMessages to true
-- don't show the search if "lock screen" is checked
if withLockScreen is "true" then lock screen
repeat with stackNumber = 1 to the number of lines ¬
in searchStack
  go to line stackNumber of searchStack
  if the result is "Cancel" then
    put empty into bkgnd field "hits"
    exit mouseUp
  end if
  if searchStack is "the stack you want to search"
  then put the short name of this stack into ¬
  searchStack
  put empty into firstFound
  repeat
    -- do the users Find command
    do findCommand
    -- quit if keyword not found
    if the result is "not found" then exit repeat
    put the foundChunk into theFoundChunk
    -- test for bug in the foundChunk
    if theFoundChunk is empty then next repeat
    -- test for full circle thru stack
    if theFoundChunk && "of" && the name of this ¬
    card is firstFound then exit repeat
    -- set marker 1st time thru stack
    if firstFound is empty then put theFoundChunk ¬
    && "of" && the name of this card into firstFound
    put the foundLine into theFoundLine
    put the foundField into theFoundField
    -- test for bug in found functions
    if theFoundLine is empty or ¬
    theFoundField is empty then next repeat
    put the visible of theFoundField into theVisible
    -- don't use this keyword occurrence
    -- if "search hidden" is checked
    if searchHidden is "true" or theVisible is "true"
    then
      add 1 to hits
      put "Hits:" && hits into msg
      -- now center the keyword in the line
      put word 2 of theFoundChunk into charToCenter
      put word 2 of theFoundLine into lineNumber
      -- calculate offset from start of the foundLine
      if lineNumber > 1 then
        subtract (the number of chars in (line 1 to ¬
```

S·T·A·X!™

more than just stacks...

STAX! is a new company dedicated to creating great HyperCard® products that help you get more out of your Mac. STAX! is more than just stacks because we design products and run STAX! *with your needs in mind.* For example, all our products have extensive on-line help, annotated scripts and no copy-protection. We don't process credit card orders and checks until we ship your product. And all upgrades will cost only $10 per disk — or less! We're dedicated to providing you with high-quality, reasonably priced solutions. So whether you're a new Macker or a long-time hacker, STAX! has products to help you work smarter.

STAX! Helper™

STAX! Helper is a three-disk set of hints, tips, buttons, scripts and stacks that makes using HyperCard as much fun as Hamburger Helper makes eating ground round a treat! There's no need to reinvent the wheel every time you build a stack. Just crack open STAX! Helper, where dozens of canned scripts are waiting to help you with sorting, finding, indexing, special effects, import/export tasks and animation routines to create screen-motion and rhythm — to name just a few choice functions available from our smorgasbord of generic scripts — all yours for the pasting.

An illustrated reference manual is included with each three disk package.

Price: $59.95.

STAX! Sound Effects Studio™

STAX! Sound Effects Studio combines a SoundCap-to-SND converter, a pitch and speed modifier, a sound library and database, an automatic sound inserter (for putting sounds in stacks without ResEdit) and more than 100 professionally recorded sound effects.

The sounds range from high-tech mechanical, such as the sound of film advancing in a camera, to a wide variety of background music, both originals and classics; from human and animal sounds, including crowds, monster, and traffic noises, to the sounds of nature found in the woods, at the seashore and from the sky. In addition, Sound Effects Studio comes with a whole phantasmagoria of category-defying tones, notes, twitters, peals, hums, squeaks, jangles, bellows and booms that no self-respecting sound effects engineer can live without.

An illustrated reference manual is included with each three-disk package.

Price: $59.95.

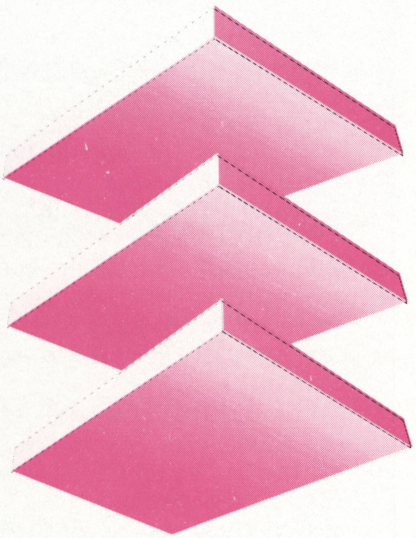
The Macintosh Bible: STAX! Edition™

"The Macintosh Bible" is widely regarded as the best and most informative book ever written about using a Macintosh. Peter Lewis of The New York Times says "('The Bible' is) like having a Macintosh expert at your side whenever you need one." The STAX!'s Edition takes it one step further and puts that Macintosh expert right inside your Mac. It's the most up-to-date information around — it even includes information from the just-released January 1988 update!

The Macintosh Bible: STAX! Edition is a three-disk set of the best hints, tips, shortcuts and Mac info the paper edition has to offer, archived electronically for fast and easy access. The STAX! Edition incorporates updates more efficiently than paper databases, allows you to create new records based on your own Mac know-how and even lets you expand and modify the records that come with it. Innovative use of the HyperCard Find command and multiple cross-referenced indexes work together to help you search effectively for information by title, subject, and class — and then go to the item or series of items you seek at the click of a mouse. The uncluttered and friendly user interface keeps all the complex relationships of the data behind the scenes and makes the STAX! Edition of "The Macintosh Bible" one of the easiest-to-navigate stacks in the business.

A paper copy of the 420-page book and an illustrated reference manual are included with each three-disk package.

Price: $79.95.



To order by credit card, call

1-800-MAC-STAX

In Texas, call 512-467-4563

S·T·A·X!
more than just stacks...

8008 Shoal Creek Blvd.
Austin, Texas 78758

Minimum system configuration for all products: Macintosh with 1Mb RAM, two 800K disk drives and a copy of HyperCard. Hard disk recommended.

VISA MasterCard

HyperCard® is a trademark of Apple Computer, Inc.

ALL STAX! products have an unconditional 30-day money-back guarantee. We don't process credit card orders and checks until we ship your product. ($5 shipping/handling charge per order.)

We also carry a complete line of HyperCard books and third-party stacks for your convenience. Ask about our low prices on books including, "The Complete HyperCard Handbook" (Danny Goodman); "HyperTalk Programming" (Dan Shafer); "HyperCard Power" (Carol Kaehler)and stacks like Focal Point, Business Class and Reports for HyperCard.

Circle 1 on Reader Service Information Card

Background Button:
Search hidden fields
From Background:
Keyword Search
AutoHilite: true
ShowName: false
Visible: true
Icon: none
Rectangle: 113,56,131,72
Style: checkBox
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: No

Background Button: Clear
From Background: Keyword Search
AutoHilite: true
ShowName: true
Visible: true
Icon: none
Rectangle: 452,67,512,84
Style: rectangle
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: Yes
See Listing 12

Background Button:
Find one occurrence
From Background:
Keyword Search
AutoHilite: true
ShowName: false
Visible: true
Icon: none
Rectangle: 261,56,279,72
Style: checkBox
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: No

Background Button: Lock screen
From Background:
Keyword Search

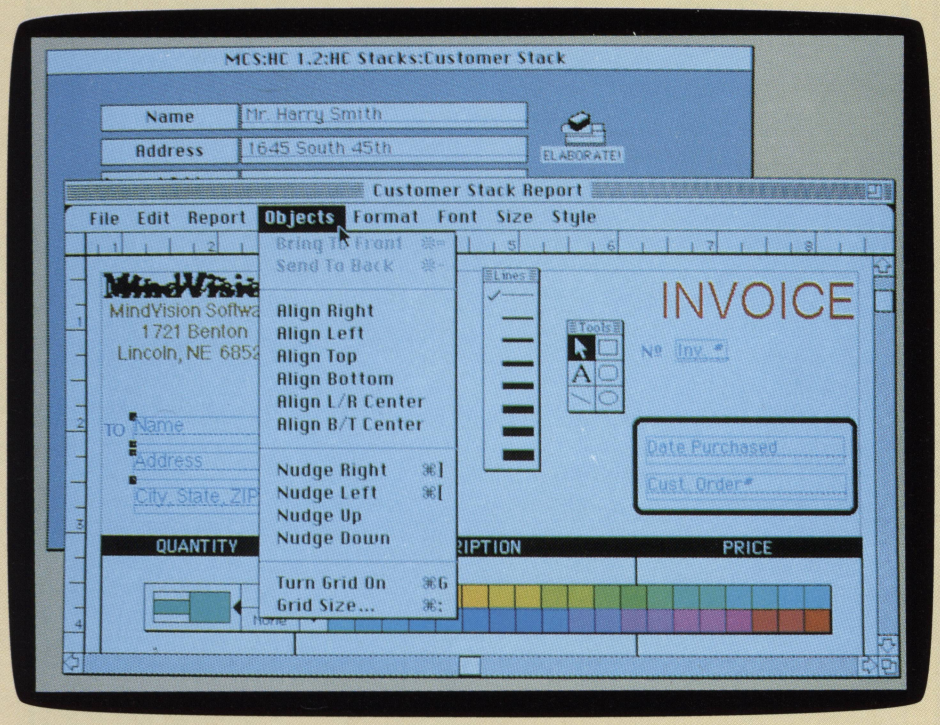
Listing 9 - Script for Background Button "Search"

```
          lineNumber - 1 of value(theFoundField))+1) ¬
          from charToCenter
        end if
        put value(theFoundLine) into searchLine
        if charToCenter > 30 then
        -- remove characters at start
          put charToCenter - 30 into toDelete
          delete char 1 to toDelete of searchLine
        else
          if charToCenter < 30 then
           -- add spaces at start
            put 30 - charToCenter into toAdd
            put char 1 to toAdd of blankString & ¬
            searchLine into searchLine
          end if
        end if
        put the length of searchLine into theLength
        -- truncate the line to fit in the window
        if theLength >= 69 then delete char 69 ¬
        to theLength of searchLine
        if theVisible is "false" then put "…" after ¬
        searchLine
        put searchLine & return after searchList
        put the long name of this card & return ¬
       after searchCard
        put theFoundChunk & return after searchChunk
        -- skip to next card if
        --"find one occurrence" is checked
        if findOne is "true" then
          if the number of cards = 1 then exit repeat
          else go to next card
        end if
      end if
      if hits >= maxHits then
        beep
        put "  ***** Maximum hits reached! ***** " ¬
        after msg
        wait 3 seconds
        exit repeat
      end if
    end repeat  -- inner loop thru one stack
    if hits >= maxHits then exit repeat
  end repeat  -- outer loop thru list of stack names
  pop card  -- return to the Keyword In Context stack
  set the lockMessages to false
  set the lockRecent to false
  openStack
   -- simulate a normal stack and card open
  openCard
  if field "stacks" is empty then put searchStack ¬
  into field "stacks"
  put searchList into field "searchList"
  if hits <> 1 then
    put hits && "hits" into field "hits"
  else
    put hits && "hit" into field "hits"
  end if
  put searchCard into field "searchCard"
  -- store foundChunk information in a hidden field
  put searchChunk into field "searchChunk"
end mouseUp
```

Why just create when you can *Elaborate!*

Elaborate!™ is the only powerful report generator that works completely within HyperCard™, allowing you to interactively switch between HyperCard and **Elaborate!** in seconds, instead of minutes as with other report generators.

Elaborate!'s unique interactive page preview mode allows you to see how your report will look when printed, before actually printing.



Elaborate! incorporates:

- Extensive selection of sorting options
- Powerful search criteria that includes 12 operators
- Calculated fields for totals and subtotals
- The ability to print relationally linked stacks
- Searching capabilites that include 12 operators
- Mail merges

And it supports color

Elaborate! allows you to create sophisticated reports in minutes with it's object drawing environment that uses floating windows for maximum flexibility. This sophisticated drawing environment, coupled with extensive search and sort criteria, allows you to print exactly what you want, how you want, and where you want it. This power is available to you without any scripting required, although access to HyperTalk is supported.

MindVision

For more information call or write:

MindVision Software
1721 Benton Street
Lincoln, NE 68521
(402) 477-3269

Ask about our introductory special for those who own Mediagenic's Reports™
Others can order **Elaborate!** before February 1, 1989 and receive 20% off!

AutoHilite: true
ShowName: false
Visible: true
Icon: none
Rectangle: 24,56,42,72
Style: checkBox
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: No

Background: Help

This background is not included in the listing, because it is mostly an explanation of what is covered in this article. If you wish to create a "Help" card, simply make a new background and place a large scrolling field on the card (be sure to name the card "Help"). Type in tips, hints, and ideas from this article that you find helpful. You can add other shortcuts and ideas of your own as you work with the stack.

Of course, the stack included on the this issue's *StackSolutions* disk contains a "Help" background and some helpful hints.

Listing 10 - Script for Background Button "Display"

```
on mouseUp
  if word 1 of field "hits" = 0 or field "hits" is ¬
  empty then
    answer ¬
    "Sorry, you need to search before displaying!"
    exit mouseUp
  end if
  put field "findCommand" into findCommand
  put field "searchChunk" into searchChunk
  put field "searchCard" into searchCard
  push card
  set the lockRecent to "true"
  set the lockMessages to "true"
  set cursor to busy
  visual effect barn door open
  -- loop through all hits in Keyword In Context field
  repeat with currentMatch = 1 to the number of lines ¬
  in searchCard
    go to line currentMatch of searchCard
    -- go to card with keyword
    if the result is "Cancel" then exit mouseUp
    select line currentMatch of searchChunk
    -- highlight keyword
    wait 1 seconds
    if the mouse is down then
      answer "What do you want to do?" with ¬
      "Continue" or "Stop here" or "End display"
      if it is "Stop here" then exit mouseUp
      if it is "End display" then exit repeat
    end if
  end repeat
  visual effect barn door close
  pop card
  set the lockMessages to false
  set the lockRecent to false
  openStack
  openCard
end mouseUp
```

Listing 11 - Script for Background Button "Export"

```
on mouseUp
  if word 1 of field "hits" = 0 or field "hits" is ¬
  empty then
    answer ¬
    "Sorry, you need to search before exporting!"
    exit mouseUp
  end if
  ask "Enter the name of the text file to export to" ¬
  with field "findCommand"
  if it is empty then exit mouseUp
  put char 1 to 31 of it into fileName
  put "Starting export to file" && fileName into msg
  open file fileName
  -- write a header with date, time,
  -- and search criteria
  write the long date && the long time & return & ¬
  "Export file for:" & return & field "findCommand" ¬
  & return & field "stackPrompt" & return & field ¬
  "stacks" & return & return & "Options:" & return ¬
```

Listing 11 - Script for Background Button "Export"

```
  to file fileName
  if the hilite of bkgnd button "Lock screen" is ¬
  "true" then
    write "Lock screen" & return to file fileName
  end if
  if the hilite of bkgnd button ¬
  "Search hidden fields" is "true" then
    write "Search hidden fields" & return to file ¬
    fileName
  end if
  if the hilite of bkgnd button "Find one occurrence" ¬
  is "true" then
    write "Find one occurrence per card" & return to ¬
    file fileName
  end if
  write return to file fileName
  put the number of lines in field "searchList" ¬
  into hits
  -- loop through all hits in the
  -- Keyword In Context field
  repeat with i = 1 to hits
    put i & "." & return into fileBuffer
    put line i of field "searchList" & return after ¬
    fileBuffer
    if last character of line i of field "searchList" ¬
    is "…" then
      put "(hidden)" into hidden
    else
      put empty into hidden
    end if
    put "Chunk:" && line i of field "searchChunk" && ¬
    hidden & return ¬
    after fileBuffer
    put "Card:" && line i of field "searchCard" & ¬
    return & return ¬
    after fileBuffer
    write fileBuffer to file fileName
    put "Export" && i/hits*100 & "% complete" into msg
  end repeat
  put "Export done" into msg
  close file fileName
  hide msg
end mouseUp
```

Listing 12 - Background Button "Clear"

```
on mouseUp
  put empty into field "findCommand"
  put "in stack" into field "stackPrompt"
  put empty into field "stacks"
  put empty into field "searchList"
  put empty into field "searchCard"
  put empty into field "searchChunk"
  put empty into field "hits"
  openCard
end mouseUp
```

If You Think HyperCard And Networking Are Not Compatible, Read On...

Networking with HyperCard–First Steps

by David G. Brader

HyperLink moved into larger offices this past summer. This was very exciting for the staff. Even more exciting, we had the opportunity to design the space because we signed a three year lease. New wiring...

New wiring of telephones, power outlets, and a built-in computer network—plugs in every wall. Networks... *HyperCard* can't be networked, right?

Wrong.

HyperCard can be made to communicate stack-to-stack over a network through a special added HyperTalk command (part of *HyperCom* from Gava).

New *HyperCard* applications such as *Focal Point II* from TenPointO (a division of Mediagenic) are making use of special XCMD's to allow E-Mail type communication through a network file server (a computer system dedicated as a central data repository for a network).

Also, *HyperCard* can be used as a front-end for networked databases via super products like the new *ORACLE for Macintosh*. More about these later.

As developers begin to use tools such as HyperCom *and* ORACLE's *Hyper*SQL/SQL*Net, the potential for new networked* HyperCard *applications is absolutely staggering.*

Step One: Getting Wired

Some people have a fear of networking. Almost all of the reasons for this fear are created because of a lack of knowledge. Did you know that the software needed to allow your Mac to communicate with another Mac is built into the machine? This same software is utilized by the Mac's system to communicate to the LaserWriter. If you use a LaserWriter, you use a network already.

The cable that connects the LaserWriter to the Mac is called an Apple *LocalTalk* cable (originally

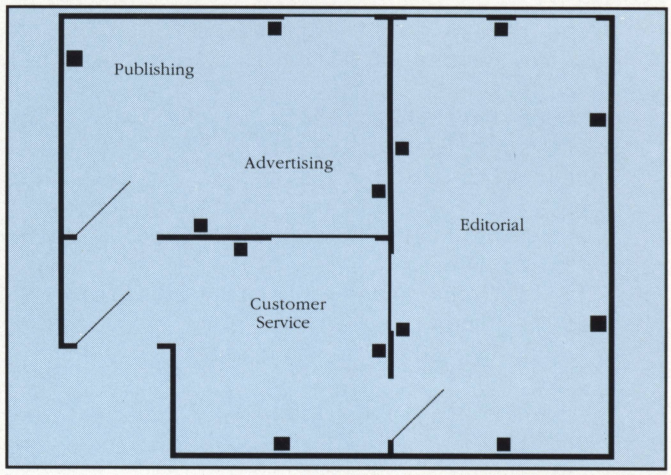


Figure 1

The location of each new PhoneNet *connection was marked on the office plan before construction started. We made sure to have a power connection near each* PhoneNet *site.*

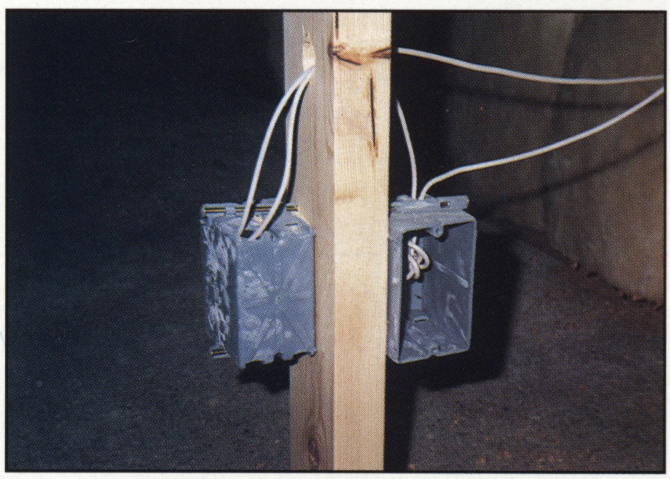


Figure 2

Two hours before the wallboard was nailed to the wall studs, the rough wiring for the PhoneNet *system was installed.*

David G. Brader serves as Publisher for HyperLink Magazine.

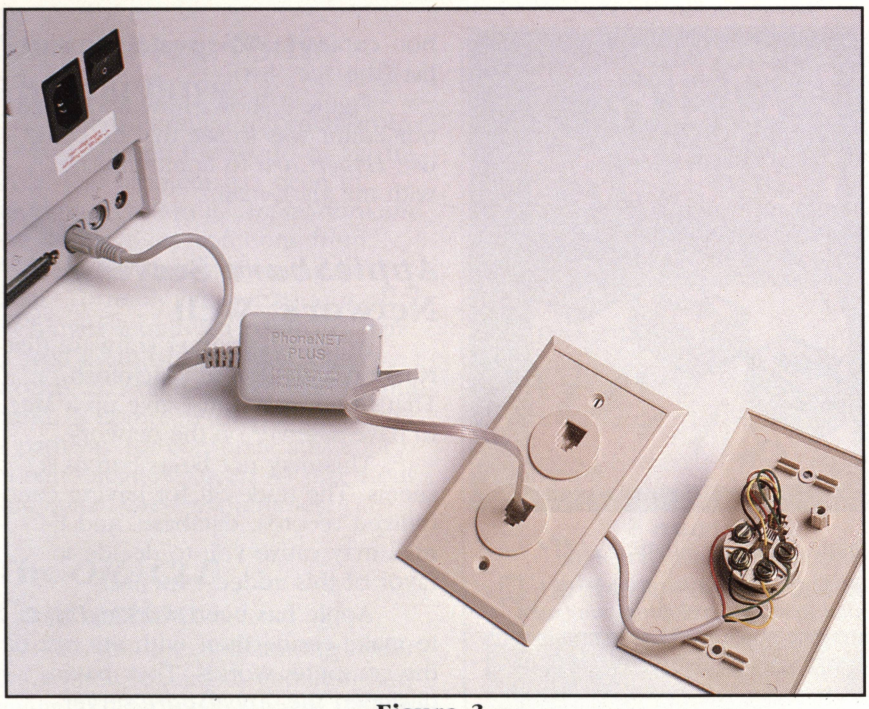


Figure 3

Here you can see the complete PhoneNet *wiring trail. Starting at the Mac's LocalTalk connector the* PhoneNet *adapter connected. The* PhoneNet *adapter connects through a normal telephone wire to a wall plug (either single or double). This plug is wired through the walls to the other wall plugs. By the way, Radio Shack stores carry the wall plugs and wire.*

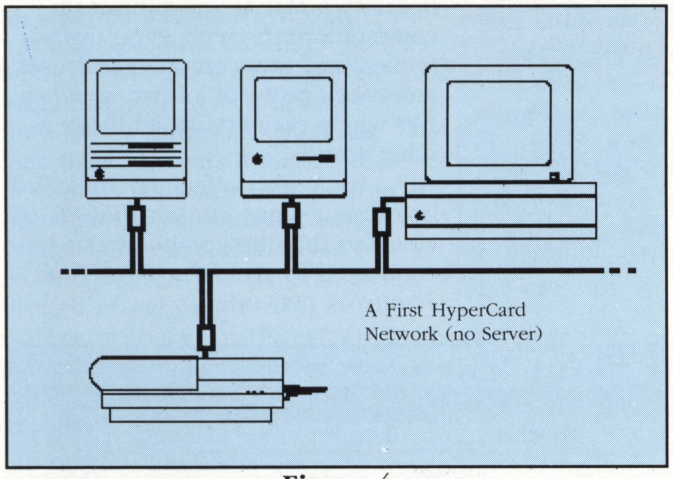


Figure 4

The first network in our offices only connected our workstations to the LaserWriter II NTX and did not allow any Mac-to Mac-communication until the HyperCom *software was installed.*

called *AppleTalk* cables). You can use *LocalTalk* cables to connect all your Macs in a network. This works fine if all the systems are in one room although the cabling can be unsightly.

Because our new office consists of several rooms, we decided to connect our network using Farallon's *PhoneNet* system. *PhoneNet* can be wired into an office using the existing office telephone wiring—maybe. If your telephone jack looks like the one in Figure 3, and has four wires as shown, you may be able to use the wiring. The trick is that your telephone system must use only two of the four wires. This leaves the other two for the *PhoneNet* network wiring.

Our existing telephone wiring used both pairs of telephone lines in our office. If you don't know how your multi-line telephone system is wired, check with the company that installed the system.

In our case, we prewired the office with a separate set of telephone lines that we dedicated to the *PhoneNet.* Because it only uses two out of the four lines, we can hook up an intercom on the spare set.

We drew up plans of the office showing a little black box for each potential location for a workstation to be on the network (see Figure 1). At the appropriate time, the rough wiring inside the new walls was installed with standard electrical outlet boxes nailed to the wall studs (see Figure 2). After the plaster and paint were dry, the wires from each electrical outlet box were connected to a telephone-style wall plug as shown in Figure 3. These phone plugs are identical to our normal telephone wall plugs, so we have them carefully marked.

Once you have the network wired, you can start sharing your printers (ImageWriters and LaserWriters) between the members of the network. With the relatively low cost of a *LocalTalk/PhoneNet* network, the sharing of printing resources is often a good enough reason to network, but let's take a look at the next step, networking with *HyperCard.*

HyperCom Tells All

At this time, the easiest way to generate networked *HyperCard* applications that we have seen comes from a software company in the Pacific Northwest called Gava. The product is called *HyperCom.* This *HyperCard* communication system is easy to install and easy to use. Each *HyperCom* software package that you purchase is licensed for a set number of computers on a network. The price for *HyperCom* varies depending on the number of computers licensed. The smallest *HyperCom* configuration costs $99.95 (two computers being the minimum).

Many forms of communication on a network require a server. *HyperCom* can be utilized to build serverless real-time *HyperCard* communication on the network.

Once *HyperCom* is installed into each networked Mac's *HyperCard,* the new HyperTalk command `tell` is available for use in scripts. *HyperCom* uses an `idle` XCMD which means that any `on idle` handlers must `pass idle` to allow *HyperCom* to work. An important feature of *HyperCom* is the ability to intercept an incoming message from another *HyperCom*-equipped Mac before allowing it to execute in your stacks.

This `ListenerOn` mode can be used in your HyperTalk scripts to limit the persons and commands that you accept from the network. The security that you develop using this listening mode will surely be put to the test by some clever network HyperJoker as he tries to surprise you with a whistling bombshell screen in the middle of your data entry. Worse still is the character that sends the following command:

```
tell them, "tell off"
```

This shuts off `them`'s *HyperCom*, isolating them from *HyperCom* network communications!

HyperCom can operate in the background under MultiFinder. Unfortunately, the current version (version 1.2) of *HyperCard* cannot. Therefore, *HyperCard* must be active on both the sender (teller) and the receiver (listener) before any direct communication is possible.

HyperCom has great potential for *HyperCard* network communications and we hope to see some exciting stacks produced using it.

The next step in network complexity is to add a server. This leads to more possibilities.

HyperMail Via *Focal Point II*

Danny Goodman has been working overtime to enhance his popular *HyperCard* product. One of the enhancements in the latest version of *Focal Point* is the addition of electronic mail or E-mail.

The simple network shown in Figure 4 will not support E-mail. E-mail requires a "Post Office" in the form of a network server. E-mail works in an analogous fashion to regular mail. The E-mail must have a destination address and be posted at the server where the E-mail is sorted and placed into the correct mail box.

Mail implies a time delay between posting and receipt. This delay implies a place for temporary storage. Thus the requirement for a network server.

Focal Point II requires that two *HyperCard* stacks be installed on the server to use E-mail. The first is the "Network Manager" stack

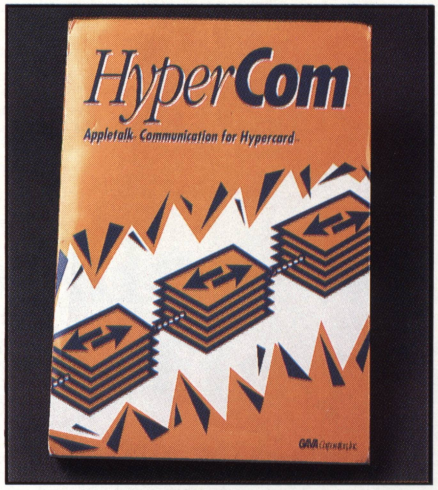


Figure 5

HyperCom *comes with a concise manual that explains everything needed to start developing your own multi-Mac* HyperCard *applications.*

which resides in its own folder. The second stack is the Mail stack that is placed in a folder inside the "Network Manager" folder. This "Mail" folder is replicated for each network *Focal Point II* user. Each of these folders carries the name of the individual as it appears in the *HyperCard* "Home" stack of the user.

Figure 6 shows the first *HyperCard* card to appear at a *Focal Point* user's Mac after pressing the E-mail button along the side of any *Focal Point II* card. An outgoing message may be sent to multiple network users (the "to" list) and carbon copies may be sent to another list (the "cc" list).

Figure 7 shows how impressed our Editor was to see that he could use *HyperCard* to communicate with me via E-mail.

AppleShare Serves A Network Well

Apple sells server software that runs on a dedicated Macintosh. That's right, you must give up a Mac to have a server on the network.

This may not be as bad as it seems. The trade-off for having centralized records, database, and E-mail may cause you to decide in favor of this added expense.

Apple has been working hard to make connections with the rest of the computer worlds. They have designed the *AppleShare* server software, the *AppleTalk* network protocol software, and *LocalTalk* wiring to work with almost all Apple computers, including Apple II family elements. By adding a special board to a Mac II you can set up communications with PC's, mainframes, and any computer that uses a network protocol known as *EtherNet* which connects to all those other worlds.

Apple also offers an *AppleTalk* board that plugs into an IBM PC or clone so that the computer can be connected as part of a *LocalTalk/PhoneNet* network.

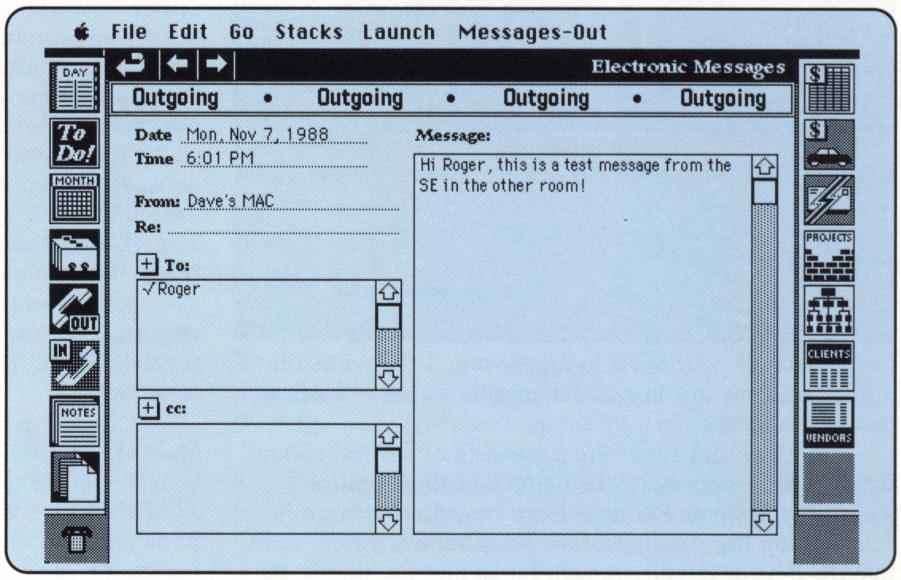


Figure 6

The new Focal Point II HyperCard *product by Danny Goodman features* HyperCard *Electronic Mail which is posted and picked up from a network server. Here I post a simple message in our Editor's mail box.*

Networking With The Big Boys

There is much information to be had beyond the world of Macintosh. The problem has been in gaining access to it. *AppleShare* and the connectivity solutions from Apple are a start at opening the door. To really blow that door off its hinges requires a company that is well established in the "other worlds" to come to take Mac by the hand. A company that could create a pathway for corporate and educational Macintosh users to reach out and touch their mainframes.

The *ORACLE* Connection

Communication among different worlds requires a common language and protocol. If these do not exist, then a trained interpreter who knows all the languages and protocols involved may serve the purpose as well.

Oracle Corporation is perfecting such an interpreter. It only needs to deal with the differences in protocol because a common language has been agreed upon. Structured Query Language (SQL) has been used for years to access data on various relational databases in the other worlds. With the advent of Hyper*SQL from *ORACLE* the Macintosh can speak the same language through *HyperCard*. [See Dan Shafer's overview of Hyper*SQL in this issue for specific information on using *ORACLE* in conjunction with *HyperCard*—Ed.]

The addition of *ORACLE's* SQL*Net to the Macintosh will allow several different network protocols to be passed with ease. SQL*Net is activated by Hyper*SQL commands written in HyperTalk script within a *HyperCard* application.

SQL*Net and Hyper*SQL are part of the new *ORACLE for Macintosh* system. This system is primarily a relational database management system (RDBMS) for local use on a Macintosh. This local RDBMS cannot be shared by more than one user either remotely or under MultiFinder locally. (At least not by direct access. Data could be locally loaded into *HyperCard* from the local database and then accessed remotely via *HyperCom.*)

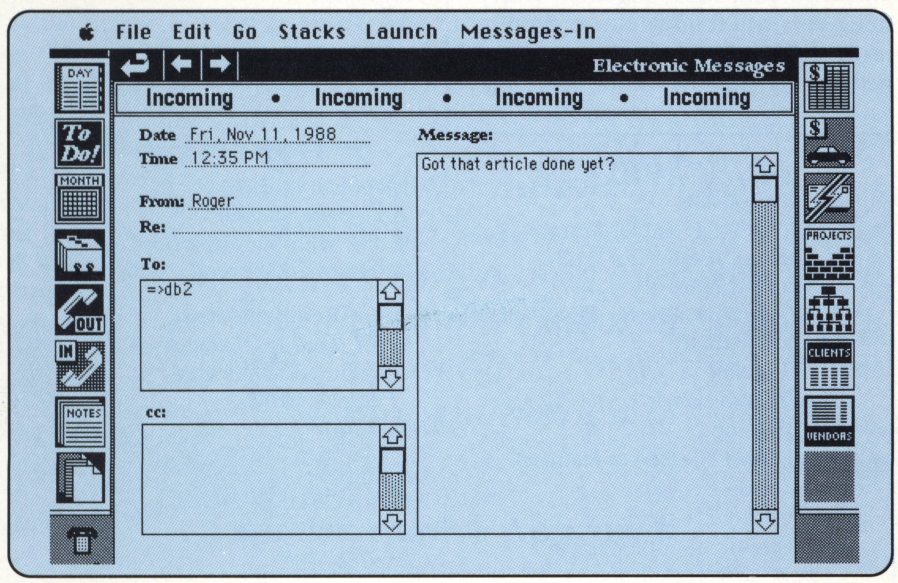


Figure 7

The reply from our Editor is just his way of gently applying his boot to my britches. (No thanks to Danny's new Focal Point II *"Incoming E-mail" stack on the server.)*

SQL*Net will eventually support the *LocalTalk, DECNet, APPC,* and *TCP/IP* network protocols. This means that in the near future your Mac will be able to access data on virtually any network—regardless of the computer or operating system.

Distributed Processing

SQL*Net's existence allows different Macintoshes to run different (or the same) *HyperCard* applications that access the same remote computer (perhaps a DEC VAX system) for the data required. With the

Macs processing away on the data, the VAX is free to run other programs or to just maintain the *ORACLE* database. This redistribution of processing to the Macintoshes can cut network traffic as well as the VAX's processing time.

Distributed Database

SQL*Net through Hyper*SQL, allows a Macintosh *HyperCard* application to access and manipulate remote *ORACLE* RDBMS data the same as local *ORACLE* RDBMS data. The data for one *HyperCard* application may be scattered over several different networked computer systems and yet, will appear as one logical database to the user.

Wow, Some First Steps!

Hey, what can I say; Networking with *HyperCard* is going to open doors from us to the rest of them. Doors that allow a view of information not possible currently on any other widely available personal computer. Views that will enhance the efforts of Macintosh users.Go ahead take that first step and wire your Macs together. Everything else will follow as you get a taste of the power of networking.

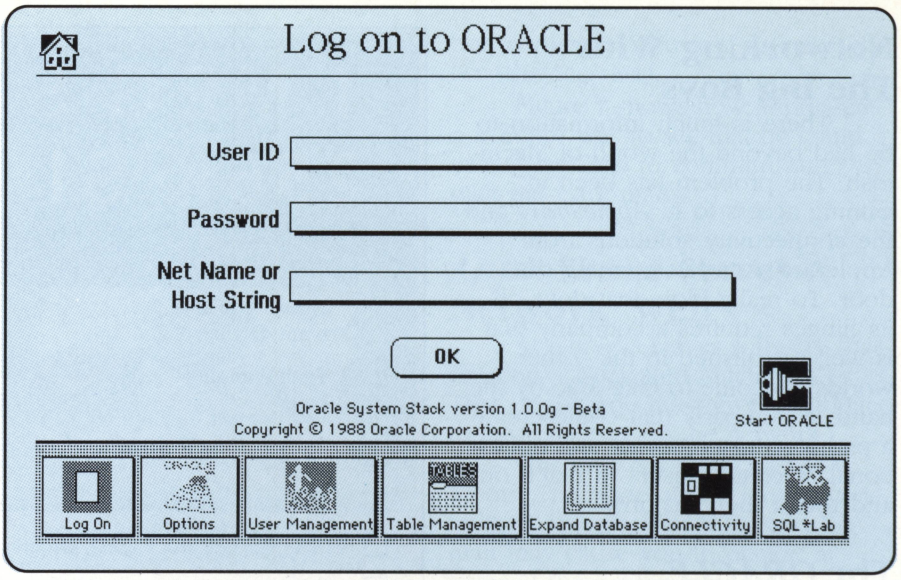


Figure 8

The startup screen from the ORACLE *System Stack that comes with* ORACLE *for Macintosh includes sound effects of a car starting when the Start* ORACLE *button is pressed. You can log on to remote* ORACLE *databases as well as your own local* ORACLE *database from this System Stack Card.*

AppleShare®

AppleShare consists of:

- 5 *AppleShare* File Server Disks:
  - Workstation installer for 512K enhanced Macs
  - Workstation installer for Mac Plus, SE, and Mac II
  - Server Installer
  - Server Administrator
  - Apple II Setup
- 3 AppleShare File Server Manuals:
  - User's Guide
  - Administrator's Guide
  - Administrator's Supplement for Apple II workstation

System Requirements:

Macintosh Plus, SE, or II with one or more hard disk drives

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

Focal Point II®

Focal Point II by Danny Goodman is in the final stages of testing as we go to press. See the overview of the product in this issue's "Product Reviews" section.

TENpointO
3885 Bohannon Drive.
Menlo Park, CA 94025
(415) 329-0800

HyperCom®

HyperCom consists of:

- A 24-page manual
- 1 disk with the *HyperCom* installer program and several example stacks.

System Requirements:

Two or more Macintosh Plus, SE, Or II computers connected via *LocalTalk/ PhoneNet* and
HyperCard 1.2

GAVA, Inc.
P.O.Box 70443
Seattle, WA 98107
(206) 784-4736

PhoneNet®

PhoneNet devices are available at your computer dealer complete with simple instructions. One unit is required for each device on the network.

Farallon Computing, Inc.
2150 Kittredge Street
Berkeley, CA 94704
(415) 849-2331

ORACLE®

ORACLE for Macintosh consists of:

- *ORACLE* RDBMS
- Hyper*SQL (HyperCard Interface)
- SQL*Net with associated protocols
- SQL* Plus®**
- Pro*C**
- *ORACLE* Call Interface
- Database Utilities
- System Stackware
- Example Stackware

System Requirements:

Macintosh Plus, SE, or II with 2 Megabytes of RAM, 5 megabytes hard disk space, floppy disk drive
HyperCard 1.2
*Macintosh Programmers Workshop*** (optional)

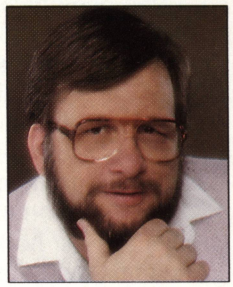
Oracle Corporation
20 Davis Drive
Belmont, CA 94002
(415) 598-8000

International Inquiries: (415) 598-8290
Telex 171437
FAX (415) 595-0630

To order *ORACLE for Macintosh* or to request further information call:
1-800-345-DBMS

** For advanced usage only

Shafer On Scripting



Hypertext in HyperCard Revisited; A Look at An Indexing Technique

by Dan Shafer

In my last column, I took a look at some techniques for creating hypertext-like links in HyperTalk. At the end of that column, I posed the question, "But what if we want the user to be able simply to point at a word and click on it?" I promised to tackle that question in this issue, so let's dive right into the subject.

You may recall that my previous efforts were flawed by the fact that they relied on the user's *selecting* some text rather than simply pointing at it and clicking once. This approach had a number of disadvantages. Among them was the fact that when the destination text which we wanted to link to appeared on a different card, we had to go through a number of programming gyrations to make up for *HyperCard's* losing track of the selection in the course of the navigation.

In attempting to solve that problem, my first impulse, which is probably like yours, is to work with the `mouseWithin` message handler that we used for the most effective of our previous efforts at achieving something resembling real hypertext. The idea is simply to detect the user's click of the mouse, identify the line on which he or she had clicked, and then find that text in the stack (in our example, the text is always in the second field on the only card in our sample stack).

The result of that attempt appears in Listing 1. As you can see, the approach is fairly straightforward. The only "tricky" part is calculating the line number on which the mouse-click takes place in a scrolling field, but that is a problem many people have solved the way we show in Listing 1.

Bugs in the Handler!

The only problem with this solution is that it doesn't always work as expected. There are at least three problems with this approach, as it turns out.

First, the fact that the first command in the handler is a `wait` command means that if the handler finds the targeted text in some other field it doesn't always let us do anything with that text. In fact, it doesn't let us do anything else at all; instead, it stays "stuck" in the `mouseWithin` handler in Field 1, with the cursor unchanging. We are in essence "locked" into this script. (This doesn't *always* happen, but it does more often than not.) Furthermore, any time the cursor re-enters Field 1, even accidentally or on its way somewhere, the handler grabs the `mouseWithin` message and traps us again. Not very elegant, after all!

The second problem is that the handler doesn't always find the text in Field 2. A fairly significant fraction of the time, it finds the text in Field 1 despite the fact that we've used two find commands. Finally, although the handler in Listing 1 is reasonably acceptable on a Mac II, running it on a Mac Plus or SE results in the system's not detecting the `mouseClick` correctly a significant portion of the time.

No amount of fiddling would get rid of these problems, so I finally brought them to the attention of Apple's Developer Technical Support team.

Although the handler in Listing 1 is reasonably acceptable on a Mac II, running it on a Mac Plus or SE results in the system not detecting the mouseClick correctly a fairly significant portion of the time.

An Alternative Approach

Chris Knepper of that group looked at the problem and proposed a more elegant solution that works just fine. His idea inspired the script that appears in Listing 2. Notice that he uses the `openField` message instead of the `mouseWithin` message. This gets around the fact that the `mouseWithin` message is sent continuously so long as the mouse remains within the boundary of an object. The `openField` message, however, is only sent once, and that solves the problem of the handler "sticking." (Thanks, Chris, for the suggestion. It pointed the way to many other approaches.)

Dan Shafer is the author of many microcomputer books, including the newly released HyperTalk Programming *(including version 1.2), and* Understanding HyperTalk *(both from Hayden Books).*

This handler works well except that it requires that the field be unlocked. The `mouseWithin` message never gets sent when the user clicks in a locked field Thus if we attempt to use this handler in a locked field nothing happens.

This turns out to be a simple problem to fix, however. Because a locked text field, unlike an unlocked field, does receive the `mouseUp` message if the user clicks in its boundaries, we can simply define a `mouseUp` handler that does the same as the `openField` handler. If we put both the `mouseUp` handler and the `openField` handler in the same field script, then it won't matter if the user or our script modifies the lockText property of the field. The hypertext links will still work as expected.

Multiple "Hits"

We have only left "one stone unturned" in our brief exploration of hypertext programming in HyperTalk in the last column and this. What if the word we want to find occurs more than once in a stack? We might like the user to be able to find the next occurrence of a word after we've found the first one for him. No problem.

In the `markText` handler, just before the two find commands are issued, place the following line:

```
put "find" && quote & it & ¬
quote into Message
```

The message box appears (you can always use a show command to make it appear off-screen if you don't want the user to see it) with the find command reproduced in it. As soon as the user hits the return key after the first find succeeds, the HyperTalk find command executes and takes him to the next location. By building in some checking for the field in which the find occurs, you could avoid a circular find loop, just informing the user that he's found the last one when a find takes place in the "Index" field.

General Thoughts

That covers the hypertext subject, at least for the moment. There are many variations and new ideas that we could experiment with, but hypertext is after all only one of many things we can do with *HyperCard* and we don't want to concentrate on any one of them for very long.

Hypertext is a very powerful idea and the somewhat limited implementations I've been discussing here should not be taken as representative of all that it can do for your users. At the same time, many of the ideas embodied in hypertext have a long way to go before realization on any machine, using any techniques. Hypertext is a rich ground for lots of future development. We undoubtedly have not heard the last of it here.

Listing 1

```
on mouseWithin
  wait until the mouseClick
  put the clickLoc into t1
  -- clickLoc stores where the user clicked
  set lockScreen to true
  -- so user doesn't see all the "find" activity
-- Calculate where click took place in terms of line
-- number in a scrolling field.
  put trunc((item 2 of t1 - top of target + scroll of ¬
   target) div textheight of target) + 1 into theLine
  -- Now use Version 1.2 function called "select line"
  -- to select the line
  select line theLine of card field 1
  -- By getting it, we store the text in the It variable
  get the selectedText
  find it
  -- find the first occurrence (where we marked it in
  -- card field 1)
  find it -- finds the next occurrence (in card field 2)
end mouseWithin
```

Listing 2

```
on openField
   markText the clickLoc
end openField

on markText t1
  set lockScreen to true
  put trunc((item 2 of t1 - top of target + scroll of ¬
  target) div textheight of target) + 1 into theLine
  select line theLine of card field 1
  get the selectedText
  find it -- find the first occurence (in card field 1)
  find it -- finds the next occurence (in card field 2)
end markText
```

The *TEX* Stack

As elegant as hypertext is, it isn't the only way to link pieces of textual information together. For many years before hypertext was even a buzz word, people employed many techniques to permit people to store and retrieve information on computers. One of the most interesting I've seen in *HyperCard* is Mark Zimmermann's intriguing and powerful *TEX* stack, a shareware product that I use all the time.

Zimmermann is an outspoken supporter of the idea that we should be sharing software and ideas to make the world a better place. In his early versions of this stack, he commented on his motivation. "I want omniscience. Until then, I build tools for people to use on massive collections of free-text information." *TEX* is indeed such a tool.

Zimmermann started out with a freeware product called *Texas*, which is still widely available. It is a great product and it's free. But he's announced he's not extending it further, and some of the goodies he's added to the product to make it valuable enough to pay a shareware fee are quite useful and powerful. To find out the latest and great-

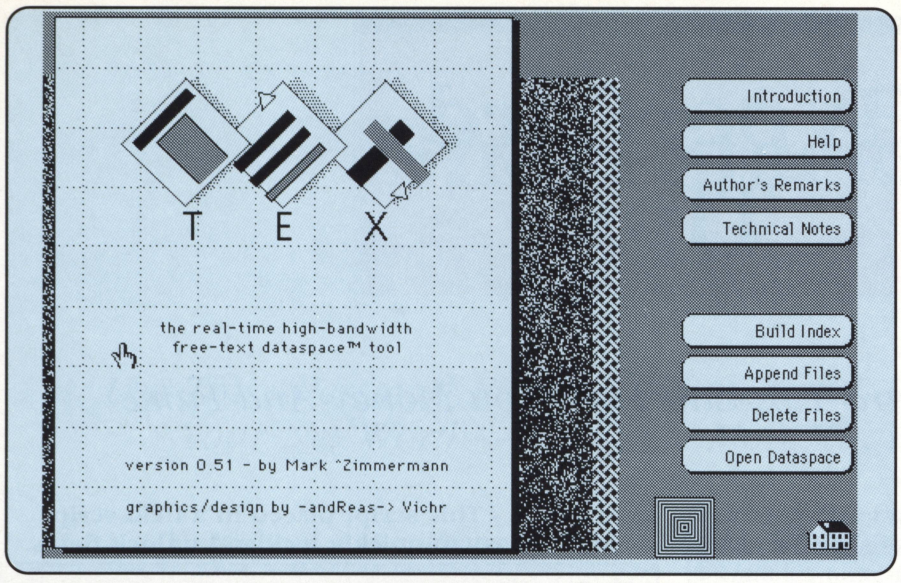


Figure 1

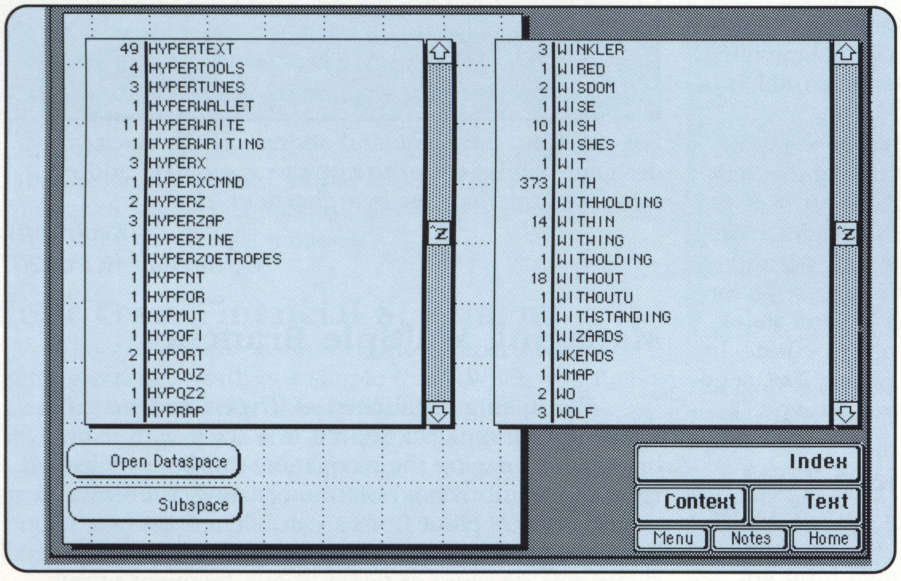


Figure 2

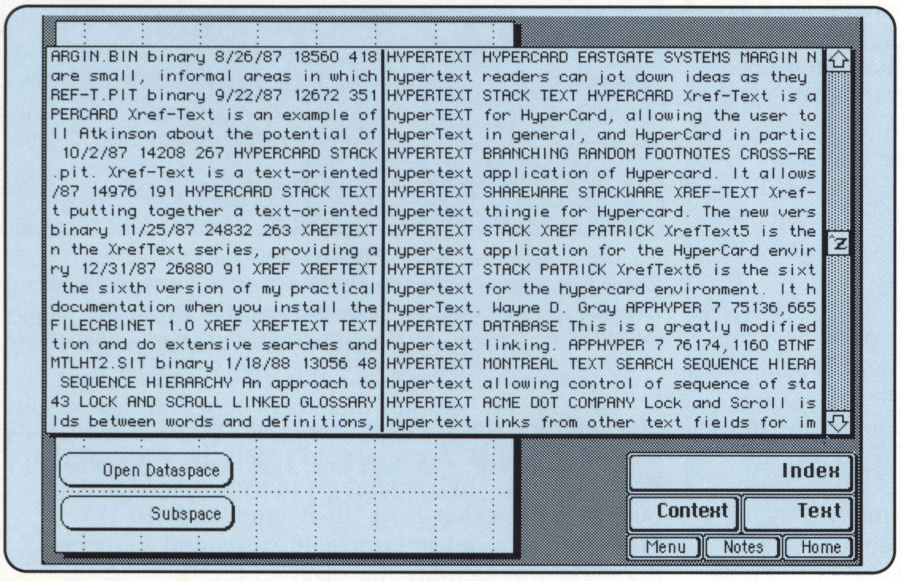


Figure 3

est about his indexing and browsing stacks, you should contact Zimmermann. He can be reached at 9511 Gwyndale Dr., Silver Spring, MD 20910. He's also prominent on CompuServe, where his ID number is 75066,2044. He asks a $10 individual, or $40 corporate, lifetime shareware fee for *TEX*. [Mark Zimmermann has allowed *HyperLink* to include the latest version of his shareware *TEX* stack on our *Stack-Solutions* disk for this issue—Ed.]

Figure 1 shows the opening screen of *TEX*. Just click on the button labeled "Build Index" and the program asks you to point it to a text-only document that you'd like it to index for you. Once you've shown it the text file, *TEX* goes off and uses some high-speed XFCN's to build a complete index to the file. It indexes every word in the file, but the process is remarkably fast.

Now that you have an index, you can open it in what *TEX* refers to as a "dataspace." When you open it, you are presented with two scrolling lists (see Figure 2). These index every word in the file and tell you how many occurrences there are of each. You can use normal Mac scrolling methods or you can click on one of Zimmermann's specially designed scroll thumbs to take an express elevator to a specific word. In Figure 2, I've asked the stack to find all references to hypertext (since that's the subject we've been covering) and the key word has scrolled to the top of the left field.

Click on the word hypertext and you are taken quickly—within 1 or 2 seconds on a Mac II—to a complete list that shows the context of all of the occurrences of the word "hypertext" (see Figure 3). Find the one that you're interested in, and *TEX* takes you to another screen where you can read the full text of the reference. If you find something that you really want to preserve for later reference, copy it to the last card in the stack, which is a free-form note field. From there, you can export or print the data.

Zimmermann challenges users to apply *TEX* to multi-megabytes of information. I've used it on a file that was 1.3 megabytes in size and the performance was far better than acceptable. Highly recommended!

HyperCard Tips

Your Hot HyperCard Tips Can Earn You Money And Fame

Change is the one constant. In this case it is *HyperLink* that is going to make a change. Prior to this issue, we had two reader-input columns called "Buttons & Bows" and "Fields of Intelligence." The submissions we received for these columns, although intriguing, were often not suitable for publication due to size, the need for XCMD's, etc. Meanwhile, we received a number of tips, and ideas which didn't fit the mold of these two columns.

In order to open the door to our readers' input, we are eliminating those columns and starting this new version of "*HyperCard* Tips" in their place. Most of the tips you see here will come from you, our readers. We'll pay from $25 to $50 for the best tips we receive, and print them for all to share, giving the author credit for the idea. We'll also add tips that the *HyperLink* staff comes up with during our stack development efforts. In addition, if you have a question about *HyperCard*, send it to us. We'll do our best to get the answer for you.

An Easy Lock & Unlock Field Tool

Popup fields present a problem to stack designer and end user alike. They are a problem for the stack designer, because these fields in their final form are usually locked, and thus uneditable. Before the stack is distributed, the stack designer will repeatedly have to unlock the field, edit the text, and then relock it. Unlocking and relocking the field is a multi-step process requiring the choosing of the field tool, selecting of the field, obtaining the info on that field, etc.

Popup fields are a problem for the end user because they often contain information that could be useful elsewhere. Yet, to copy the text, the user unlocks the field, copies the text, and then relocks the field to put it back the way it was.

The elegant little script in Listing 1 can be placed into any popup field to eliminate these problems; the `else` statement can contain any other `mouseUp` handler you might want (in this case just `hide me`).

To edit or copy the text in the popup field, all the stack designer or user has to do is to hold down the Command key and click on the field to unlock it, do the editing, then hold down the Command key again, and click on the field to lock it. I propose that stack designers start using this scripting convention to allow easy access to the normally locked text in their stacks.

Listing 1 - This script placed in a field script makes it easy to quickly lock and unlock fields.

```
on mouseUp
  if the cmdKey is down
  then set lockText of me to not ¬
  lockText of me
  else hide me
end mouseUp
```

For example, the name and address of the stack designer will be easier to copy to a separate address rolodex if this handler is in the field.

Kevin Altis
Columbia, MO 65201

Managing Multiple Branches

The linking capabilities of *HyperCard* are great for hypertext applications, but in a stack with many branches managing the navigation can be a tricky job. Here's a useful technique for facilitating a return to a correct branch point from a card shared by two branches in a stack with intersecting branches. Figure 1 shows this structure as taken from a fragment of my

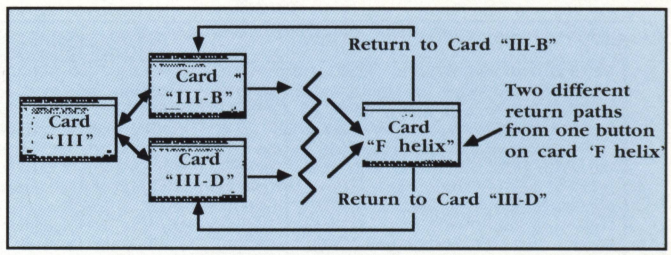


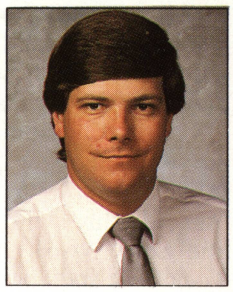
Figure 1

There is a single "Return to Outline" button on card "F helix." Depending upon how the user has reached the card, a different script is "written" by the "Diagram" button on either card "III-B" or "III-D" ensuring a return to the correct card.

biochemistry stack called "Hemoglobin, Gas Transport and Allosterism." A user could reach the card on the right, "F helix," by clicking on pieces of bold text or buttons from either card "III-B" and card "III-D." (There are several intermediate cards and branches not shown to simplify the diagram.)

—continued on page 51

Short Stacks



Use This "Personal Financial Statement" Stack to Calculate Your Net Worth—It May Be More Than You Thought

by William K. Balthrop

How much are you really worth? Unless you have recently applied for a loan you probably don't know the answer. This issue's "Short Stack" provides you with a *HyperCard* tool that allows you to track your net worth on the computer—to make periodic updates any time you wish. Net worth is calculated by subtracting all of your liabilities from all of your assets. Liabilities are debts and money owed. The loan on your car or house is a liability. Anything you own, even if you still owe on it, is an asset.

Your Own Financial Statement

The *HyperCard* version of the standard form for figuring out net worth is a simple form. When you first open the stack you see the screen divided into two halves (See Figure 1). The left half is for assets, the right is for liabilities. Each line represents a major category for assets or liabilities. In addition, each category may contain several detailed records of individual assets or liabilities.

All objects and graphics reside at the card level of a single card. Be sure the User Level is set to Scripting in your "Home" stack. Select "New Stack..." from the File menu, and be sure that "Copy Current Background" is not checked so the background is left blank.

Start by entering the bold lines as illustrated in Figure 1. To get the bold line, tear off the tool palette, and place it in a convenient location on your screen. Then double click on the Line tool and select the third line from the left. The following text headings are entered by using the Text tool (the big A on the palette). I chose 14 point New York for Assets, Liabilities, Total Assets, and Total Liabilities. Then, using 18 point New York bold, put "Net Worth" at the bottom.

Next, create the objects listed below. Do these in order so the different fields and buttons are layered properly. Several fields and buttons must cover others as the stack is being used, and entering them in the correct order is the easiest way to ensure this.

William "Kelly" Balthrop is the Director of R & D at HyperLink and has been developing and publishing computer software for the last decade.

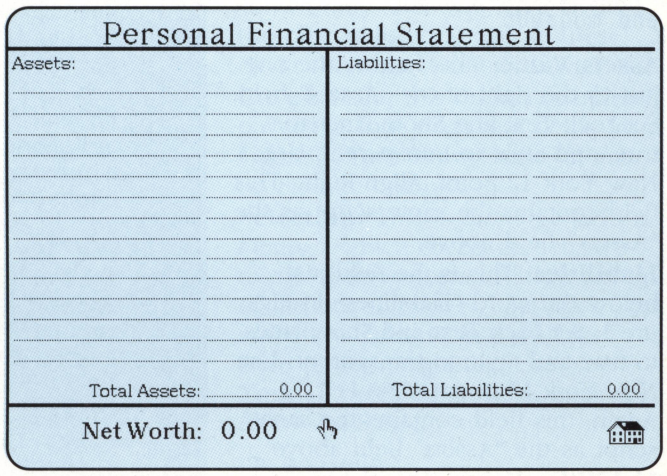


Figure 1

This card is your personal financial statement. Assets are listed on the left, with liabilities to the right. All graphics, buttons, and fields are on the card level. The background is left blank.

Top Header: Create this field at the very top of the screen as shown in Figure 1. Do this by choosing "New Field" from the Objects menu. Next, open the "Field Info..." Dialog box (just double-click on the newly created field) and set Lock Text to true, and Style to

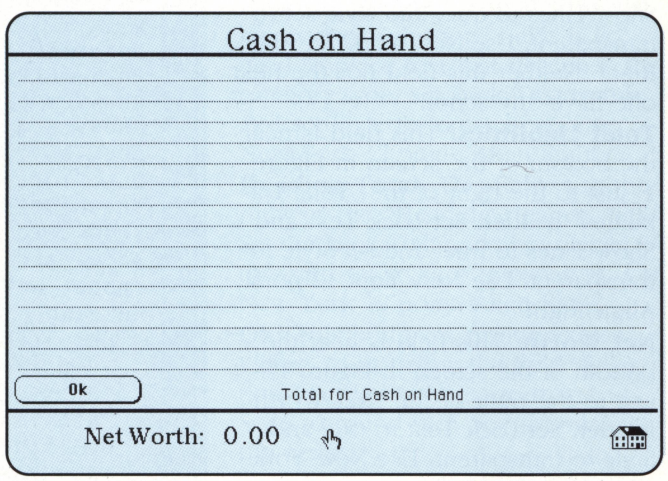


Figure 2

This is the detail screen. It is actually the same card as that shown in Figure 1, with several normally invisible fields and a button covering the other fields.

opaque. Be sure to name this field "Top Header," and name all subsequent fields and buttons as designated, or the scripts will not work properly. Click on the "Font" button selecting New York 24 point and Align Center.

Assets: This is the field just below the word "Assets" in Figure 1. Set Lock Text and Show Lines to true and Style to transparent. Use New York 12 point, Align Left. Open the field's script editor and enter the following script:

```
on mouseUp
  ChangeStatement
end mouseUp
```

Assets Value: This field is the one just to the right of the "Assets" field. Set Lock Text and Show Lines to true, and Style to transparent. Use New York 12 point, Align Right. This field contains the same script as the "Assets" field above.

Liabilities: This is the field just below the word "Liabilities" in Figure 1. Set Lock Text and Show Lines to true and Style to transparent. Use New York 12 point, Align Left. Again, this field contains the same script as the "Assets" field above.

Liabilities Value: This field is just to the right of the "Liabilities" field. Set Lock Text and Show Lines to true and Style to transparent. Use New York 12 point, Align Right. This field also contains the same script as the "Assets."

Total Assets: This field is near the bottom of the screen in Figure 1, just to the right of the "Total Assets" header. Set Lock Text and Show Lines to true, and style set to transparent. Use New York 12, Align Right. Neither this field nor the next one have any script.

Total Liabilities: This field is near the bottom of the screen in Figure 1, just to the right of the "Total Liabilities" header. Set Lock Text and Show Lines to true, and set Style to transparent. Use New York 12 point, Align Right.

Net Worth: This field is near the bottom of the screen in Figure 1, just to the right of the "Net Worth" header. Set Lock Text to true and Style to transparent. Use New York 18 point, Align Left.

Home: The "Home" button resides in the lower-right corner of the screen. I turn off Show Name, set the

Listing 1 - Stack Script for "Personal Financial Statement"

```
on openStack
  hide menubar
  put "Personal Financial Statement" into card field ¬
  "Top Header"
end openStack

on ChangeStatement
  global Heading,Type,Pline
  put the clickLoc into Position
  put trunc((item 2 of position - 33) / 16) into Pline
  if item 1 of position < 254 then
    put "Assets" into Type
  else put "Liabilities" into Type
  if line Pline of card field Type is empty then
    ask "Name of Heading:"
    set cursor to watch
    put it into Heading
    if Heading is empty then exit ChangeStatement
    put Heading into line Pline of card field Type
    set lockscreen to true
    doMenu "New Field"
    set name of last card field to Heading
    hide card field Heading
    choose browse tool
  else
    put "Do what with this" && Type && "Category" ¬
    into Query
    answer Query with "Cancel" or "Delete" or "Change"
    set cursor to watch
    Put line Pline of card field Type into Heading
    if it is "Cancel" then exit ChangeStatement
    if it is "Delete" then
      set lockscreen to true
      choose field tool
      show card field Heading
      select card field Heading
      doMenu "Cut Field"
      choose browse tool
      delete line Pline of card field Type
      delete line Pline of card field (Type && "Value")
      TotalValues
      exit ChangeStatement
    end if
  end if
  put empty into card field "Selection"
  put empty into card field "Selection Value"
  put empty into card field "Heading Total"
  put Heading into card field "Top Header"
  show button "Cover"
  show card field "Selection"
  show card field "Selection Value"
  show button "Ok"
  put "Total for " && Heading into card field ¬
  "Heading Header"
  show card field "Heading Header"
  show card field "Heading Total"
  repeat with x=1 to the number of lines of card field ¬
  Heading
    put item 1 of line x of card field Heading ¬
    into line x of card field "Selection"
    put item 2 of line x of card field Heading ¬
    into line x of card field "Selection Value"
    put line Pline of card field (Type && "Value") ¬
    into card field "Heading Total"
  end repeat
end ChangeStatement

on ChangeSelection
```

Listing 1 - Stack Script Continued

```
global Type
  get the clickLoc
  put trunc((item 2 of it - 19) / 16) into Sline
  if line Sline of card field "Selection" is empty
  then
    ask "Item Name:"
    if it is empty then exit changeSelection
    put it into Iname
    put Iname into line Sline of card field "Selection"
  else
    put "Do what with these" && Type & "?" into Query
    answer Query with "Cancel" or "Delete" or "Change"
    if it is "Cancel" then exit ChangeSelection
    if it is "Delete" then
      delete line Sline of card field "Selection"
      subtract line Sline of card field ¬
      "Selection Value" from card field "Heading Total"
      delete line Sline of card field "Selection Value"
      exit ChangeSelection
    else put line Sline of card field "Selection" ¬
    into Iname
  end if
  put "Value of" && Iname & ":" into Prompt
  put GetNumber(Prompt,line Sline of card field ¬
  "Selection Value") into line Sline of card field ¬
  "Selection Value"
  set cursor to watch
  put 0 into Total
  repeat with x=1 to the number of lines of card field ¬
  "Selection Value"
    add line x of card field "Selection Value" to Total
  end repeat
  put CheckNumber(Total) into card field "Heading Total"
end ChangeSelection

on TotalValues
  global Type
  put 0 into Total
  repeat with x=1 to the number of lines of card field ¬
  Type
    add line x of card field (Type && "Value") to Total
  end repeat
  put checkNumber(Total) into card field ("Total" && ¬
  Type)
  put checkNumber(card field "Total Assets"-card field ¬
  "Total Liabilities") into card field "Net Worth"
end TotalValues

Function GetNumber Prompt,Default
  repeat
    ask prompt with default
    if it is empty then put default into it
    put CheckNumber(it) into NumberOut
    if NumberOut is "Error" then next repeat
    else
      return NumberOut
      exit repeat
    end if
  end repeat
end GetNumber

Function CheckNumber Num
  Set the numberformat to "0.00"
  put 0 into decimals
  put 0 into minuses
  repeat with curNum = 1 to length(Num)
    if char curNum of Num = "."
    then put decimals + 1 into decimals
```

Style to transparent, and use icon 1011 "Home." Here's the script:

```
on mouseUp
  go home
end mouseUp
```

The Second Layer

Figure 2 looks like a different card, but it's the same one shown in Figure 1. The fields and buttons shown in Figure 2 are hidden when they are not in use. They are shown when a user chooses a category, from Figure 1, and they cover up the fields shown in Figure 1. Create the following objects in the order shown only after you have created the objects listed above so these cover the ones defined above.

Cover: Make this button completely cover the four main text fields from Figure 1. Turn Show Name off and set Style to opaque.

Selection: This is the large field in Figure 2 which covers the left two-thirds of the screen, and sits on top of the button "Cover." Set Lock Text and Show Lines to true with the Style set to transparent. Use New York 12 point, Align Left. Enter the following Script:

```
on mouseUp
  ChangeSelection
end mouseUp
```

Selection Value: This field also sits on top of the button "Cover" and is just to the right of the field "Selection." Set Lock Text and Show Lines to true and the Style to transparent. Use New York 12 point, Align Right. The script is the same as for card field "Selection."

Heading Header: This field is just below the "Selection" field. It is used to place text that describes the category type. Set Lock Text to true and set Style to transparent. Use New York 12 point, Align Right.

Heading Total: This field is to the right of the "Heading Header" field. Set the Lock Text and Show Lines to true and set style to transparent. Use New York 12 point, Align Right.

Ok: This button is positioned on Figure 2 just below the field "Selection." Set Show Name to true and Style to round rect. See listing 2 for the script.

Now you have all of the buttons and fields set up, it's time enter the stack script shown in Listing 1.

Listing 1 - Stack Script Continued

```
  if char curNum of Num = "-"
    then put minuses + 1 into minuses
    if char curNum of Num is in "-1234567890." and ¬
    decimals < 2 and minuses < 2 then next repeat
    else
      return "error"
      exit CheckNumber
    end if
  end repeat
  if Num = "." or Num = "-" or ¬
  Num = "-." or Num = ".-" then
    return "error"
    exit CheckNumber
  end if
  add zero to Num
  return Num
end CheckNumber
```

Listing 2 - Button Script "Ok"

```
on mouseUp
  global Heading,Type,Pline
  set cursor to watch
  put card field "Heading Total" into ¬
  line Pline of card field (Type && "Value")
  put empty into card field Heading
  repeat with x=1 to the number of lines of card field ¬
  "Selection"
    put line x of card field "Selection" into ¬
    item 1 of line x of card field Heading
    put line x of card field "Selection Value" into ¬
    item 2 of line x of card field Heading
  end repeat
  put card field "Heading Total" into ¬
  line Pline of card field (Type && "Value")
  hide card field "Selection"
  hide card field "Selection Value"
  hide button "Ok"
  hide card field "Heading Header"
  hide card field "Heading Total"
  hide button "Cover"
  put "Personal Financial Statement" into card field ¬
  "Top Header"
  TotalValues
end mouseUp
```

Assets:

Cash on Hand
- Savings
- Checking

Stocks & Bonds
- Stocks
- IRA

Real Estate
- Home
- Rental Prop.

Accounts Receivable
- For work on house

Notes & Contracts
- Land sales contract

Personal
- Furnishings
- Tools

Autos
- Car

Liabilities:

Accounts Payable
- Due to subcontractor
- Grocery tab at store

Notes & Contracts
- Loan from uncle Fred

Mortgages
- Home
- Rental Prop.
- Commercial Prop.

Bank Loans
- Car
- Furniture
- Tools

Figure 3

Examples of categories and detail items that can be used in your financial statement.

This script contains most of the handlers. This allows several fields to share a single routine. Before you try to use this stack, you must hide all of the fields and buttons shown in Figure 2. This is easy to do from the Message box. Type Command-M to show the Message box, then type

```
hide button "Cover"
```

and press Return. Hide the following fields and buttons in a similar manner:

```
card field "Selection"
card field "Selection Value"
card field "Heading Header"
card field "Heading Total"
button "Ok"
```

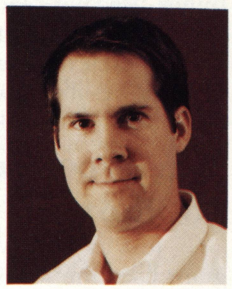
Using The Stack

To use this stack, click on a line to either list your assets on the left side of the screen or your liabilities on the right side. Ask boxes prompt you for category titles, item names, and amounts. When you click on a blank line on the screen shown in Figure 1 you are prompted for a category name. After entering the name, the detail screen shown in Figure 2 appears. Your category is displayed at the top of the screen. Click on a line to be prompted for the item name. When you click "Ok" in this Ask box, you are then prompted for the amount. The error routine we are using to check numeric input only allows for digits, decimal points, and minus signs. Therefore, if you try to enter commas, dollar signs, or any other non-numeric value, the Ask box will reappear until you enter a correct numeric value.

When you click the "Ok" button at the bottom of the screen, the screen in Figure 1 appears with your new category. Totals are automatically accumulated, and your net worth is displayed. If you click on an item or category, you are prompted as to whether you wish to delete it, change the amount, or cancel and do nothing.

I hope you find this stack both useful and instructive. Feel free to add to it and modify it to suit your own needs. Let us at *HyperLink* know about any creative innovations that you come up with.

Xpanding HyperCard



The HyperCard Toolbox And The Glue Routines

by James Paul

In the previous "Xpanding HyperCard" column, we compared HyperTalk to Pascal. We demonstrated two versions of the same routine, and noticed how much faster the Pascal external routine ran than the HyperTalk equivalent. An XCMD can be much faster, but can it do everything that a HyperTalk script can? The answer is an emphatic, "Yes!" The key to an external routine's ability to communicate with *HyperCard* is a set of Pascal (or C) routines that I like to call the "*HyperCard* Toolbox." In this issue, we'll look at a simple XFCN called "SortIt," and examine how *HyperCard* Toolbox routines are used.

The SortIt XFCN

Because we're going to concentrate on how the *HyperCard* Toolbox works, we're just going to give a basic overview of the SortIt XFCN. It employs one of the simplest methods for alphabetical sorting called the "bubble sort." No bathtubs are involved, but it employs an analogy that "heavier" numbers sink and "lighter" ones rise to the top. The weight of an object depends upon what is known as its ASCII value. (ASCII stands for American Standard Code for Information Interchange.) In this code, the letter "A" is "lighter" than the letter "Z," and a "1" is "lighter" than a "9." We sort a field of lines, that is, a Return character (ASCII value 13) is used as the character that separates, or "delimits" as programmers call it, the different pieces of information to be sorted. The SortIt XFCN compares pairs of lines, starting at the beginning. Whenever we find a pair that is heavier on top (i.e., before a lighter one) we swap the pair, sinking the heavier one down. We do this to the entire list several times, until no more changes need to be made.

Of course, we could do this in a script fairly easily, but the XFCN version will perform much more quickly. We can use this to gain insight into how the *HyperCard* Toolbox works. There is one limitation you should be aware of if you get the *StackSolutions* disk containing this XFCN—no line to be sorted can exceed 255 characters. This is because Pascal deals with strings of characters that may be no longer than 255 characters. There are ways of overcoming this limitation in Pascal, but for simplicity's sake, I have restricted each line to be a normal Pascal string. If one of the lines in the field to be sorted is longer than 255 characters, the line is shortened by throwing away any characters beyond the 255th.

James Paul is chief programmer for Paul Software Engineering, and he is the author of Icon Factory *from Hyperpress. He is also the author of numerous XCMD's and XFCN's, which include DoList, ListRes, SReplace, Pad, and SortIt.*

Each HyperCard *Toolbox routine is accessed by using what is commonly called a "glue routine."*

What is a Toolbox?

Most of us have heard the term "Macintosh Toolbox," but what exactly does this term mean? No, it's not supposed to imply that we need to keep some screwdrivers and a wrench handy while we program. The Mac Toolbox is described in detail in *Inside Macintosh* Volumes I through V, published by Addison Wesley.

A Toolbox is a set of routines that is used as a set of building blocks by programmers. The Macintosh Toolbox is the set of routines contained in Read Only Memory (ROM) of the Macintosh, and in its System file. These Toolbox routines are available to every program that runs on the Mac, and this availability is greatly responsible for the large amount of consistency in Mac software. With the powerful building blocks in the Mac Toolbox, programmers can give programs that do completely different things a similar look and feel. Imagine placing two children on the floor with some toy blocks. Each child can make a tower very different from the other child, but these towers look and feel similar because they are both made with the same type of blocks.

The Macintosh Toolbox has blocks for building windows, dialog boxes, menus, and other Mac-like

things. Because each Macintosh program is made out of these blocks, each one has the look and feel that we all know so well.

The HyperCard Toolbox

Just as each Macintosh program uses the Macintosh Toolbox, each *HyperCard* XCMD or XFCN can use the *HyperCard* Toolbox. Each *HyperCard* Toolbox routine is accessed by using what is commonly called a "glue routine." These glue routines are simply the procedure and function definitions that we need for the particular language that we are using. When we need to use a Toolbox routine (either Mac or *HyperCard*) we just call our glue routine. Glue routines stick our code and the Toolbox's code together!

There are too many *HyperCard* Toolbox routines to list and describe all of the glue routines in this article. For full details and other information, pick up a copy of Gary Bond's book, *XCMD's for HyperCard.*

Virtually all XCMD's or XFCN's glue routines communicate with *HyperCard* via an important programming device which is the the parameter block.

The Parameter Block

When an XCMD or XFCN is called, *HyperCard* passes a value called a "pointer" that points to this parameter block. This pointer actually points to a block of memory that *HyperCard* prepares for any XCMD. Being an equal-opportunity application, *HyperCard* doesn't care in which language the external routine was written, and passes the *same* pointer regardless of whether we write our XCMD with Pascal, C, or Assembly. So, although we'll be giving examples in Pascal, keep in mind that we could be using another compiled language with different syntax just as easily.

What do we do with this special pointer that *HyperCard* gave us? Take a look at the parameter block as it is defined as a "Pascal record" (see Listing 1). A Pascal record is a group of variables that all stay together. Each of the names

Listing 1 - Pascal Record of The Parameter Block

```
XCMDBlock =
RECORD
  paramCount:   Integer;
  params:       ARRAY [1..16] OF Handle;
  returnValue:  Handle;
  passFlag:     Boolean;
  entryPoint:   ProcPtr;
  request:      Integer;
  result:       Integer;
  inArgs:       ARRAY [1..8] OF Longint;
  outArgs:      ARRAY [1..4] OF Longint;
END;
```

Listing 2 - Pascal of The EvalExpr Glue Routine

```
FUNCTION EvalExpr(paramPtr:XCMDPtr; expr:Str255): Handle;
BEGIN
 WITH ParamPtr^ DO
  BEGIN
    inArgs[1]:=ORD(@expr);
    request:=xreqEvalExpr;
    DoJsr(entryPoint);
    EvalExpr:=Handle(outArgs[1]);
  END;
END;
```

in the left column followed by a colon stands for a particular variable that our external routine and *HyperCard* each access to aid in communication. The first two (paramcount and params) let us get any parameters our routine requires. The third (returnValue) allows us to send a result value back to *HyperCard* after we're done using our external routine.

What can we do with the other six variables in the parameter block? Number four (passFlag) tells *HyperCard* whether to pass the handler name (our XCMD name) up the *HyperCard* hierarchy, just as the `pass` command in a regular script does. The remaining five variables are specifically for communicating with what I've called the *HyperCard* Toolbox.

Here's a quick rundown on what each of the other parameters are for:

- entryPoint: This is a pointer that points to a special place inside *HyperCard. HyperCard* Toolbox routines use this pointer to know where to go when they need to ask *HyperCard* to do things.
- request: This is a number that tells *HyperCard* which Toolbox routine is calling it. Each Toolbox routine has its own number.
- result: This is a number that tells if any problem occurred when *HyperCard* executed the Toolbox request.
- inArgs: This is an array of 8 numbers. Each Toolbox routine uses this to send *HyperCard* the actual text or numbers to be worked with.
- outArgs: This is an array of 4 numbers. Each Toolbox routine uses this to get text or numbers back from *HyperCard* when needed.

The EvalExpr Routine

In the SortIt XFCN, we only use a couple *HyperCard* Toolbox routines, but one of them is perhaps the most flexible routine in the Toolbox. The EvalExpr routine evaluates an expression. This sounds fancy, but you've probably done it many times from the Message box. If you type "card field 1" into the Message box, you'll get the contents of card field 1. (Notice, that the card must contain a card field.) There, you've just evaluated an expression. EvalExpr does exactly

the same thing. This lets our XCMD do anything we can do from the Message box. (In fact, if you can't do it from the Message box, you can't do it with EvalExpr, either.) So, let's evaluate.

The SortIt XFCN uses EvalExpr (I pronounce it Evil-Exper) to get the contents of a field to sort. We could just pass the contents of the field as a parameter, and skip calling EvalExpr, but again, what would become of our example? Actually, by passing the name of the field itself to our XFCN, we have a way to find out things about the field besides the contents using, you guessed it, EvalExpr. Listing 2 shows the Pascal definition for the EvalExpr glue routine.

Now that we know what the parameter block variables are used for, we can see how EvalExpr works. We pass it the parameter block pointer *HyperCard* gave us along with the expression we want to evaluate. The EvalExpr glue routine takes over preparations for talking to *HyperCard* by stuffing our expression into an "inArgs" variable. Next, the number of the *HyperCard* Toolbox routine is placed into the "request" variable. This number is defined as a constant (it happens to be *HyperCard* Toolbox routine number 2) at the same time that the glue routines are defined.

After we set up these two parameter block variables, the glue routine jumps to the special spot inside *HyperCard* that's pointed to by "entryPoint." *HyperCard* will do its stuff, evaluate the expression, then hop back to the next line of the glue routine, which gets the answer from the "outArgs" variable. The glue routine quits, returning a handle to the result.

If we want to make sure that *HyperCard* didn't have any trouble digesting our expression, we can check the "result" variable in the parameter block after we are done calling EvalExpr. If "result" is anything but zero, *HyperCard* didn't like what it tried to swallow.

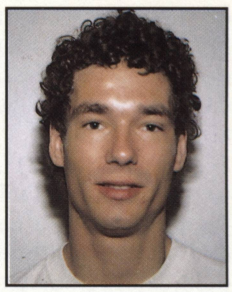
We could do each of the steps in the glue routine ourselves each time we wanted to use a *HyperCard* Toolbox routine, but using the glue routines is much more efficient. It looks better too. Those are the basics of how the *HyperCard* Toolbox is used to communicate between our XCMD's and *HyperCard*. So far there are 29 routines in the *HyperCard* Toolbox. Each one is useful in a different way, and each uses the same methods to get its job done.

The complete source code and stack for the SortIt XFCN is on this issue's *StackSolutions* disk. Also, for you programmers, or adventurous people who would like to dabble in Pascal, the routine is written so you can change the delimiter to any character you want—like a comma to get the routine to sort by items instead of lines. You'll need to have a Pascal compiler (I used the TML compiler for this XFCN), but the change is easy for even a relatively inexperienced Pascal programmer. Be sure to send us your ideas for XCMD's and XFCN's you'd like to see!

Here's how to contact me:

- c/o *HyperLink Magazine*, P.O. Box 7723, Eugene, OR 97401
- Compuserve 72767,3436
- GEnie J.PAUL

HyperCard Haiku



The Medium Is The Message

by Rhett Savage

As any 21st-century school child knows, behind every action in *HyperCard* is a message. When an object receives a message, it first looks at its own script to see if it has a handler for the message. If such a handler is found then it is executed, but if is not found then the message continues "up the inheritance path" (from buttons and fields, to card, to background, to stack, to "Home" stack, to *HyperCard*) until it finds an object which will act on it.

Messages all obey the same rules, but in a sense there are two kinds: those that are sent to your objects automatically by the *HyperCard* environment (system messages) and those you send yourself (both HyperTalk Commands and other messages defined in handlers or XCMD's).

A HyperTalk programmer's task is to adapt to, and direct, the flow of messages, both of which can be done in interesting ways. What follows are scripting examples designed to show how the message flow may be most effectively used. These include a few of the more powerful "tricks of the trade." If any of the ideas presented below are unfamiliar, a close study of them should make you a much better scripter.

System Messages

System messages are a built-in part of the *HyperCard* environment—comparable to a human's autonomic and sensory nervous system. System messages are sent automatically to objects in response to user actions or events in the *HyperCard* environment, from `mouseUp` to `openCard` to `idle`. Answering such messages is the way stacks "do their thing," so a working knowledge of the many system messages is essential. Each can be used in a unique way to obtain otherwise inaccessible information about what is happening in and around your stacks. System messages are a stack's high-fidelity link to the environment. The handlers in Listings 1, 2, and 3 illustrate how to trap the system messages `openStack`, `closeStack`, and `closeField`.

Rhett Savage is a part-time HyperCard *consultant living in Portland, Oregon. He can often be found wandering the aisles of Powell's Books.*

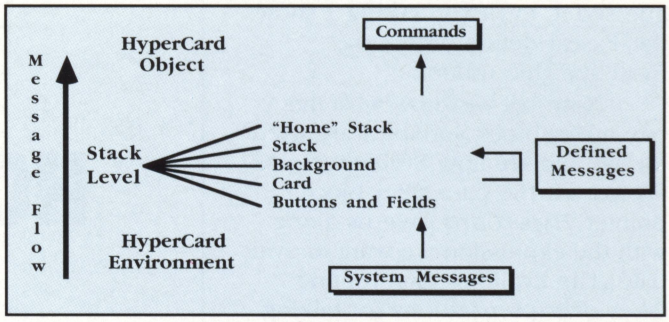


Figure 1

Listing 1

```
on openStack    -- just an example...
  global theirLevel, currentScore
  if the version < 1.2 then
    answer ¬
    "This stack needs HyperCard 1.2." ¬
    with "naturally"
    go HOME
  -- exit to HOME if they have
  -- old version of HC
  end if
  put the userLevel into theirLevel
  -- save to restore later
  set userlevel to 3
-- prevent tampering with your stacks?
  put  field "savedScore" of card ¬
  "scoreCard" into currentScore
  -- initiate a global variable from
  -- a stored value; a useful idea!
  hide the msg
  pass openStack
end openStack
```

When a stack is first opened the `openStack` message is sent to the stack. This is the stack's opportunity to make sure that the *HyperCard* environment is set up properly for it, and perhaps even to begin doing something. A typical example is the handler in Listing 1 that does a number of simple HyperTalk operations.

This works in conjunction with a `closeStack` handler to restore the original `userLevel`. The example in Listing 2 also checks to see if the stack would be benefited considerably by compaction. If it would, the option to compact is presented to the user.

Listing 2

```
on closeStack
  global theirLevel, currentScore
  set userLevel to theirLevel
  -- back the way it was...
  put currentScore into field ¬
  "savedScore" of card "scoreCard"
  -- ditto
  if the freeSize of this stack > 50000 ¬
  then
    answer "Compact the stack?" with ¬
    "cancel" or "yes"
    if it is "yes" then doMenu ¬
    "compact stack"
  end if
end closeStack
```

Listing 3

```
on closeField
  -- a typical use of "closeField",
  -- and a simple but  intricate handler!
  -- the repeat loop waits for a correct
  -- answer or a "cancel" selection.
  put me into myDate
  repeat until myDate is empty
    convert myDate to abbreviated date
    if the result is empty
    then exit repeat
    -- correctly formatted date
    ask ¬
    "Date was not formated correctly" ¬
    with myDate
    put it into myDate
  end repeat
  put myDate into me
end closefield
```

Listing 4

```
on mouseUp
  -- a single "mouseUp" handler for a
  -- whole stack...
  put the short name of the target ¬
  into buttonHit
  if buttonHit is "next"
  then go next card
  if buttonHit is "prev"
  then go prev card
  if buttonHit is "data input"
  then doMyInput  -- etc.
  -- etc.
end mouseUp
```

Listing 5

```
on mouseUp  -- in the card/stack script
  if the name of the target contains ¬
  "button"
  then go card (the short name of ¬
  the target)
  -- this handler does all the work
  -- for an indefinite number
  -- of buttons, as long as each button
  -- is named for the card it links to!
end mouseUp
```

Another very useful system message is `closeField` which is sent to a field upon closing it if its contents have changed. This message allows stacks to look at and react to what the user has put into a field. Listing 3 is a typical `closeField` handler, as well as an example of compressed HyperTalk code. It makes a field smart enough to know that it should contain a date, and to prompt the user with a series of Ask boxes if necessary.

Of course, examples could go on indefinitely. The point is that we should learn to work with the full range of system messages.

Intercepting Messages Late & The "target" Function

System messages follow the usual *HyperCard* inheritance path until they encounter an appropriate handler. For example, if a button is clicked on that has no `mouseUp` handler, a `mouseUp` message will find its way to the the card, the background, and the stack. Such messages can be trapped by handlers at these levels, and acted upon.

This possibility is especially useful in combination with the important HyperTalk function `target`. This function returns the object that *originally* received the current message. This allows, for example, an `on mouseUp` handler at the stack level to learn which button was actually clicked on.

So, in theory you could use one huge `on mouseUp` handler in your stack script, which consists of many `if…then` statements which imparts a separate behavior to each button in the stack (see Listing 4).

This is, in some respects, like other computer languages (where a whole program is one continuous stream of text), but it is generally not good style in HyperTalk. The obvious exception is when using a single `on mouseUp` handler for more than one button results in a great increase in efficiency. This might happen, for example, if the names of the buttons work directly with the handler.

To take a simple example, suppose a given card has many buttons with names. Suppose that each button links the user to a card with the same name as the button. The "Jane" button leads to a card named "Jane," and so on. The crude way to do this is is to give each of the buttons an `on mouseUp` handler that takes the user to a particular card:

```
Button "Jane"
on mouseUp
  -- script of one of many similar buttons
  go card "Jane"
end mouseUp
```

Rather than having individual scripts for buttons like this, it would be far better to have a single `mouseUp` handler in the card script. This technique allows the messages sent to the various buttons to pass through to the card layer as shown in Listing 5.

You can apply this technique in many ways by letting system messages pass up the hierarchy and use a general handler employing the `target` function to find out where each message starts. If the object names are

well selected, very fine HyperTalk results. Danny Goodman cleverly employs it in his second book, *The HyperCard Developer's Guide*, in a section called "The Ultimate Handler Reduction."

Commands & Messages That You Define

System messages are sent automatically, but your scripts send their own messages all the time. In fact, you are sending a message with every line of HyperTalk code. If you say `go Home` then you are sending the message `go` up the message path. Like any message it looks for a handler in each object it encounters, starting with the one whose script sent the message. If it finds no handlers (or is intercepted by a handler and "passed" along), eventually it passes all the way up to *HyperCard* itself. At this point, if it is a command from the HyperTalk language it is executed. Usually command messages pass through to *HyperCard*, but it is quite possible to trap them with a handler.

Of course, besides HyperTalk commands you may also send your own messages, defined by you in message handlers. Such a handler starts with the name of the message: `on [message name]`. This is how in HyperTalk we create what are called "subroutines" or "procedures" in other languages.

For example, suppose there is a melody that plays at different points in your stack (see Listing 6). One way to implement this is to simply include this `play` command in the handlers requiring it as you write your stack. Because it is only one line it might be the best approach, but there are drawbacks.

- You must remember the line in order to type it in
- Your stack will be larger than necessary because this line is duplicated in several places

The more efficient alternative is to define the new message in Listing 7. With this handler at the stack level, you can always produce the melody just by sending the message `playNotes`.

```
answer "play music?" with ¬
"yes" or "no"
if it is "yes" then playNotes
```

Listing 6

```
play "Harpsichord" tempo 380 ¬
"b3 c#4 d e f# g a#3 g4 f# e d c# d r f# r b r c# r b r a# r bw"
```

Listing 7

```
on playNotes
  play "Harpsichord" tempo 380 ¬
  "b3 c#4 d e f# g a#3 g4 f# e d c# d r f# r b r c# r b r a# r bw"
end playNotes
```

Listing 8

```
function clickLine
  return (trunc(((scroll of the target) + (item 2 of ¬
  the clickLoc) - (item 2 of the rect of the target)) ¬
  div the textHeight of the target) + 1 )
end clickLine
```

The benefits of this structuring your code are greatest when longer sequences are involved.

Defined Messages Are Like New Commands

Messages defined by handlers are potentially very powerful. When you define a message with a handler, for most practical purposes you are adding a new command to HyperTalk (in that part of Hyperspace which is beneath your handler in the inheritance path). Like any command, this one is invoked by sending a message up the path. The message is understood and resolved higher up, either by a handler or (in the case of a HyperTalk command) by *HyperCard* itself.

Messages you define yourself may take the form of functions as well as commands. (A function is like any other message handler, except that it returns a value when it is called.) To demonstrate the potential power of user defined commands, Listing 8 shows a beautiful handler which "extends" HyperTalk in a natural direction.

This function handler does several simple calculations to allow fields to easily report which line of themselves has been clicked on by the user. When `clickLine()` is called by a field which has been clicked on, it returns the number of the most recently clicked on line. This is extremely useful, because it allows multiple choices to be presented to the user in a scrolling field, and lets the user select by clicking on a choice. Such a procedure is a standard part of the Macintosh user interface, but no access to it is provided in HyperTalk. This function originally appeared in *Windoid #3*, Apple's own *HyperCard* user group newsletter, and was a fix by Apple employee Brian McGhie of the original handler written by Gary Bond.

For the curious, here is how the calculation works, step by step: To find which line of a scrolling field was clicked on, first find how far down in the field the click, measured in pixels, happened. This is equal to the number of pixels from the top of the screen to the click (item 2 of the `clickLoc`) minus the distance from the top of the screen to the top of the field (item 2 of the `rect` of the `target`). To this add the number of pixels which are part of the field but currently scrolled off the screen (the `scroll` of the `target`). This is how many pixels down in the field the click was, and we can easily find the actual line number by dividing the total pixels by the number of pixels in each line (the `textheight` of the `target`). The result will have to be truncated to make it into an integer, and to get the actual line number we need to round it up by adding 1.

Note that `clickLine` can be used by any field because the `target` function returns the object which first received the current sys-

tem message. The `clickLine()` function is thus a powerful extension of HyperTalk. For example, if `clickLine` is in the stack (or "Home" stack) script, this handler could go into any locked field:

```
on mouseDown
  put "you clicked on line ¬
  number" && clickLine() ¬
  && "of me."
end mouseDown
```

This handler puts a cute message into the message box. A much more powerful use is the following: Suppose you have a stack (like the standard *HyperCard* "Address" stack) that consists of a series of cards with a distinct identifying text in some field of each card—e.g., the person's name. In order to browse through the stack a user normally looks through the cards one by one or jumps through them using the `find` command. It would be excellent to provide the user with an overview of the stack (in the form of a list) letting her jump to whatever card she selects. The `clickLine()` function makes it amazingly easy.

An Easily Improved "Address" Stack

If you add the `mouseUp` handler in Listing 9 to the background layer of the "Address" stack, it will compile a list of the names for all the cards in the stack. A corresponding background field called "listField" must be created with the `mouseDown` handler in Listing 9 at the same time.

You must add the `clickLine` function to the background script or above, and that's all there is to it. Whenever the user pushes the new button, a scrolling field appears containing a list of all the names from the stack. When the user clicks on the field, it disappears, and the user is transported to the card matching the name clicked on.

Intercepting Command Messages

Because commands sent to *HyperCard* are really messages, they can be trapped in the usual way by a handler at any point in their path. There are several reasons that this might be useful. You might want to

block certain operations, or perhaps handle them in a particular way.

A simple example is the following handler: It blocks all `sort` commands that reach it from other handlers or from the Message box:

```
on sort
  answer ¬
  "sorting not allowed." ¬
  with "understood"
end sort
```

The "pass" & "send" Commands

One very important HyperTalk command is `pass`. When you issue the statement `pass [message name]`, execution of the current handler is ended, and the named message passes on up the object hierarchy. It is used when you have trapped a message that can also be handled further up the message stream. When a handler is ready to release control, it "passes" the message up the stream. This command allows us to selectively trap messages, act on them (when it is desired), and then release them (where it is appropriate). For example, the following handler executes the `sort` command only if the user insists.

```
on sort
  -- at the stack level
  answer ¬
  "sorting not recommended."¬
  with "SORT" or "okay"
  if it is "SORT"
  then pass sort
end sort
```

Or, suppose you wanted to stop users of a stack from using the `set` command to increase their `userlevel`. The handler in Listing 10 would take care of this nicely.

The final example is similar. Whenever the user (or a handler) makes a choice from the *HyperCard* menu bar, a `doMenu` message is sent to the current card. This message can be trapped by a handler so that any menu selection can be monitored by your stack. The handler in Listing 11 traps all menu selections, and blocks those where the user is trying to delete a card. In that case it asks permission before going ahead with the delete.

Listing 9

```
on mouseUp
   set cursor to "watch"
   put empty into field "listField"
   repeat with i = 1 to the number of cards
      put first line of field "name and address" of ¬
      card i into line i of myNames
   end repeat
   put myNames into field "listField"
   show field "listField"
end mouseUp

on mouseDown
   hide me
   visual effect wipe right to white
   visual effect wipe left
   go card clickLine()
end mouseDown
```

Listing 10

```
on set param1, param2
  if "userlevel" is in (param1 & param2)
  then answer "Do not reset userlevel!" with "okay!"
  else pass set
end set
```

Listing 11

```
on doMenu theChoice
  if theChoice is "delete card"
    then
      answer "are you sure?" ¬
      with "delete" or "no thanks"
      if it is "no thanks" then exit "doMenu"
    end if
    pass doMenu
  end if
end doMenu
```

I hope you see that redefining HyperTalk commands might come in handy. A final point worth mentioning is that if you are writing handlers to trap commands sent by the user, you might inadvertently get in the way of your own scripts. For example, what about a stack with the the `doMenu` handler just given? The same stack might also have a handler that occasionally deletes cards from the stack. It must do this by sending a command message to *HyperCard*, but how can such a message get past past the `doMenu` handler which we wrote to trap user commands?

A good solution is the HyperTalk command `send [message name] to [object name]`. This command allows messages to travel directly to specified objects, and is therefore useful for bypassing parts of the hierarchy. For example, in the given situation a handler could delete a card by saying:

```
send "doMenu delete card" ¬
to HyperCard
```

This statement sends a message that is not trapped by an `on doMenu` handler at the stack level.

Finally

HyperCard messages are a broad, shallow subject, but they are also "the right stuff." I have taught HyperTalk programming to many people, and imparting a few central insights to them seems to make all the difference. It is these insights that I am trying to introduce in this column. It is safe to say, however, that if any of the examples are unclear, your scripting will improve through study of them.

HyperCard Tips
—continued from page 38

On the "F helix" card there is a single button called "Return to Outline" that returns the user to the Outline card from which the user began. One way to implement this would be to use the `push card` command at the first branch, and put a `pop card` command in the "Return to Outline" button on the "F helix" card. In stacks with myriad branching opportunities, however, this approach could lead to some cards never being popped. This is bad form and can lead to memory problems. I've devised a method of simply having the "Diagram" button on each of the Outline cards "re-write" the script of the "Return to Outline" button.

When the user presses the "Diagram" button on card "III-B," the script in Listing 2 "writes" a script (see the put command in the sixth line of Listing 2) and puts it into card button id 1 ("Return to Outline") on card "F helix." But when the user presses the "Diagram" button on card "III-D," the script in Listing 3 "writes" a different script which it puts into that same button.

Thus no matter how the user gets to the "F helix" card, this method ensures a return to the correct Outline card.

Listing 2 - This script for card button"Diagram" on card "III-B"

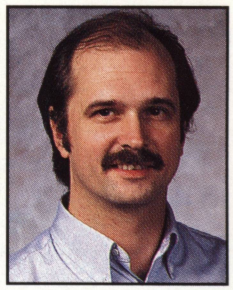
```
on mouseUp
  set the lockScreen to true
  go to card "F helix"
  hide the message box
  set script of card button id 1 to empty
  put "on mouseUp" & return & "go to card" && quote & ¬
  "III-B" & quote & return & "end mouseUp" into ¬
  scriptHolder
  set the script of card button id 1 to scriptholder
  hide the message box
  go to card "Heme"
end mouseUp
```

Listing 3 - This script for card button "Diagram" on card "III-D"

```
on mouseUp
  -- insert the same first four lines as above
  put "on mouseUp" & return & "go to card" && quote & ¬
  "III-D" & quote & return & "end mouseUp" into ¬
  scriptHolder
  set the script of card button id 1 to scriptHolder
  hide the message box
  go to card "Distal Histidine"
end mouseUp
```

James Baggott
Yeadon, PA 19050

A Look at The Present State of...



HyperCard And CD-ROM

by Roger Wood

One of *HyperCard's* big strengths is its ability to organize and search large amounts of data. The strength of Compact Disc (CD) Read Only Memory (ROM) is storing large amounts (over 500 megabytes per disc) of data. These two products appear to be a natural marriage of software and hardware.

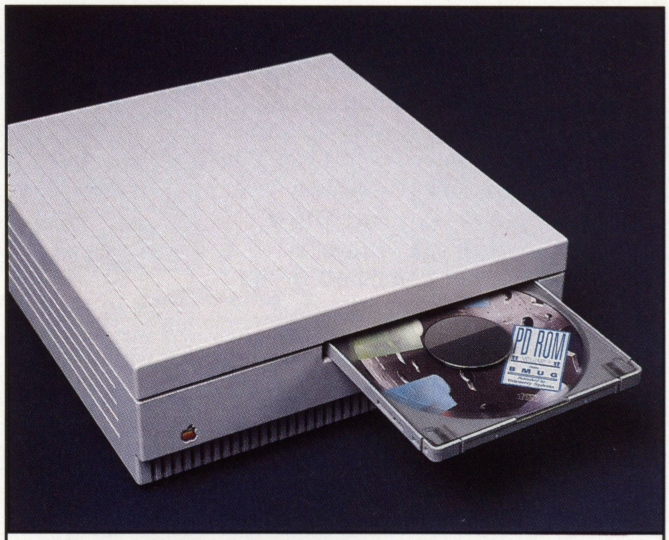
Yet, the reviews of the *AppleCD SC* (CD for Compact Disc, and SC for SCSI interface) have been mixed. Some hail it as a breakthrough for Apple—as an opportunity to turn the CD ROM industry from promise to reality. Others say the drive is too slow for managing the large amounts of data, and too expensive.

The criticism regarding expense is somewhat valid because there is still a limited amount of software available on CD—but this is already beginning to change. Databases ranging from the *MEDLINE Knowledge Finder,* Aries System Corporation, phone 617-689-9334, a subscription-based service for physicians, to a number of art and photographic databases now appearing on the market. Thus, for people with specific needs, the new *AppleCD SC* can be a valuable investment. The to-be-released Grolier's *Electronic Encyclopedia* (phone 201-947-9839), and Highlighted Data's *Merriam-Webster's Ninth Collegiate Dictionary* (phone 703-241-1180) are representative reference works that make CD ROM an exciting storage medium.

It's Big, But Slow

As to the slowness. If you are used to a fast access hard disk, the slow performance is a definite disappointment. The access time is often compared to using floppies on an unexpanded Mac Plus. But when you consider the alternative to swapping several hundred floppies or employing banks of hard disks to access 550 megabytes of data the slowness is both understandable and forgivable. The slow access is also due to a certain extent to a limitation of existing search software. Several new search engines (including Xiphias's Xearch XCMD discussed below) are adding speed. Coupled with careful data management, these new search techniques can make CD access time quite tolerable.

Roger Wood is the Editor of HyperLink Magazine.



Product Name:	*AppleCD SC*
Company Info:	Apple Computer 20525 Mariani Avenue Cupertino, CA 95014 (408) 996-1010
Price:	$1199

Networking A CD ROM

An almost obvious use of CD ROM is to employ it on a network. (Be sure to read "Networking with *HyperCard*—First Steps" in this issue for the basics on putting together a network employing *HyperCard.*) If only one person in a company has access to that much information, it may not be worth the investment for the CD player or some of the large databases CD ROM makes possible—but if dozens of people can grab the files, it begins to be more attractive.

We installed the system on our *AppleShare* network here at *HyperLink*, and we were generally pleased (see Figure 1). The process is much like installing any hard disk volume on the server. The only addition is to put the CD software drivers into the system folder of both the server's system and the server's Administration disk's system. Also, any time you add a new volume to the server, each user must use the Chooser to gain access to that volume.

There is one caveat: Only one CD can be on the network during any session. That is, you must shut down and reinstall the network each time you want to change

discs. Because, when *Appleshare* is installed, it must make a directory of every file and folder on a disk, the installation of some CD's can be lengthy. On the positive side, once a disc with several files has been installed, access time of opening files and getting directories actually improves. Once the server has all of the file and folder names in memory, searches are much more rapid. This was pointed out by Raines Cohen of BMUG which brings us to...

Product Name:
BMUG PD ROM Volume I
Company Info:
BMUG
1442A Walnut St. #62
Berkeley, CA 94709-1496
(415) 549-BMUG
Price: $100

BMUG has made the plunge early by creating this CD-ROM of Public Domain (PD) software. Actually, it is "Publicly Distributable" software because much of the software here is shareware—and virtually all of the classic Mac shareware and public domain files are here. The Software index, shown in Figure 2, is a *HyperCard* stack (a major effort in itself) containing 7206 cards—each referencing a different Mac file. That implies there are over 7000 different pieces of software on this disk—a huge compilation. The scrolling field on the right organizes the types of software as

> BMUG Newsletters, Browsing Tools, Business, Desk Accessories, Demos, Education, FKEYs, Fonts, Games, Graphics, Mac II, Messages, Pictures, Programming, Sounds, Stacks, Telecom, Text, and Utilities.

Each of these categories represents a folder full of more folders, which are full of files. If your user group already has a collection of PD disks from BMUG, the files on the *PD ROM* are familiar to you. If your user group doesn't have a collection the *PD ROM* is probably the best way to maintain such a library of files. Normally disks containing these files cost from $3 to $10 each from various sources. To get the full complement of software presented here on floppies could cost more than the $1300 investment to buy the CD ROM player and the *PD ROM* disc. A *HyperCard* user gets the benefit of seeing how BMUG chose to handle the indexing, because the Index is written in *HyperCard.*

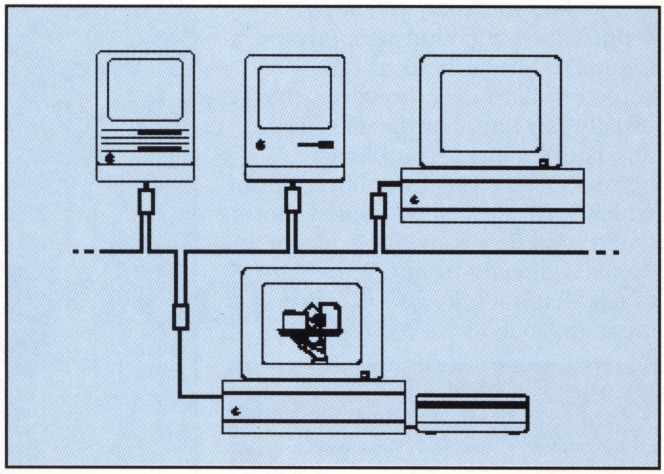


Figure 1
This diagram shows part of the current network configuration at HyperLink Magazine. *The CD ROM disc loaded into the player must be registered as a server volume, and connot be changed without restarting the server.*

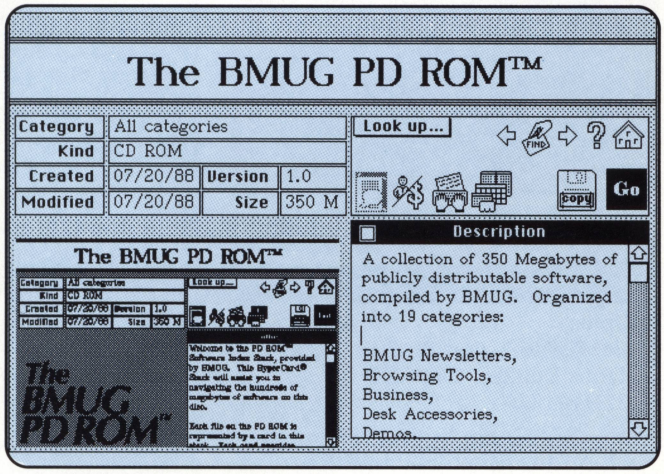


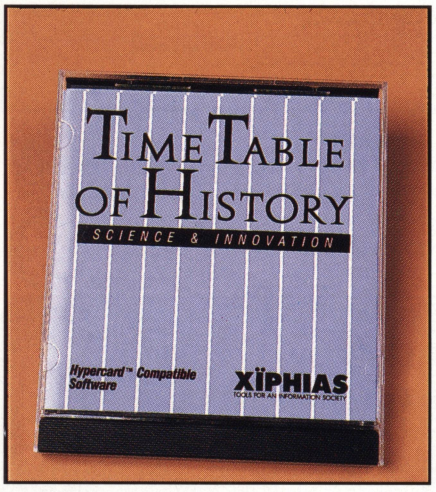
Figure 2

The product is not without drawbacks. One has to do with a limitation of the Macintosh Finder. Trying to open too many folders (and there are plenty on the *PD ROM*) can cause the system to freeze up. If you launch applications from *HyperCard* or use a desk accessory Finder substitute like *DiskTop* (from CE software) this problem can be avoided. Also, not all programs on the disk work under the latest Apple system, so bombs and system lock-ups could happen—not really the fault of BMUG, more the nature of new systems. As long as the user appreciates these possibilities, and knows that a simple reset of the system is all it takes to get started again, these present no great problem. All in all the value of the disc easily exceeds the $100 price tag.

Developing Stacks for CD ROM

Some difficulties in running stacks on this disc reveal a very important part of *HyperCard*/CD ROM development. In our first session of browsing, we found a few stacks that gave "Never heard of background button..." error messages. Upon investigation we discovered that the scripts were trying to reset properties, such as the name of a background button, but CD ROM

is a locked medium, and such scripts don't function properly on a CD ROM. When these stacks were copied to hard disk, however, they usually ran fine. For the *PD ROM*, this is not a major problem, because it is basically a library from which software can be copied. But if you're creating something meant to be used directly from CD, be sure to test that it works on a locked medium, such as a locked floppy.



Product Name:
Time Table of Science and Innovation
Company Info:
Xiphias
13464 Washington Blvd.
Marina Del Rey, CA 90292
(213) 821-0074
Price: $150

An even earlier *HyperCard* pioneer of the CD ROM medium is the Xiphias company. Their *Time Table of History* (the *Science and Innovation* disc is the only one available at this writing) makes unique use of *HyperCard* and CD ROM. It combines a graphic representation of a Time Line (Figure 3), Time Table cards (Figure 4), and spoken audio which work together endeavoring to make history less of a linear experience by linking events across time.

You can begin looking for information by choosing either the Time Line or the Time Table. The Time Line identifies many key periods of history (like when Napoleon ruled), and also shows the rise and fall of certain technological influences (like steam power). Clicking on an area of the Time Line plays an audio spoken message of an important technological event that occurred at that time. Clicking

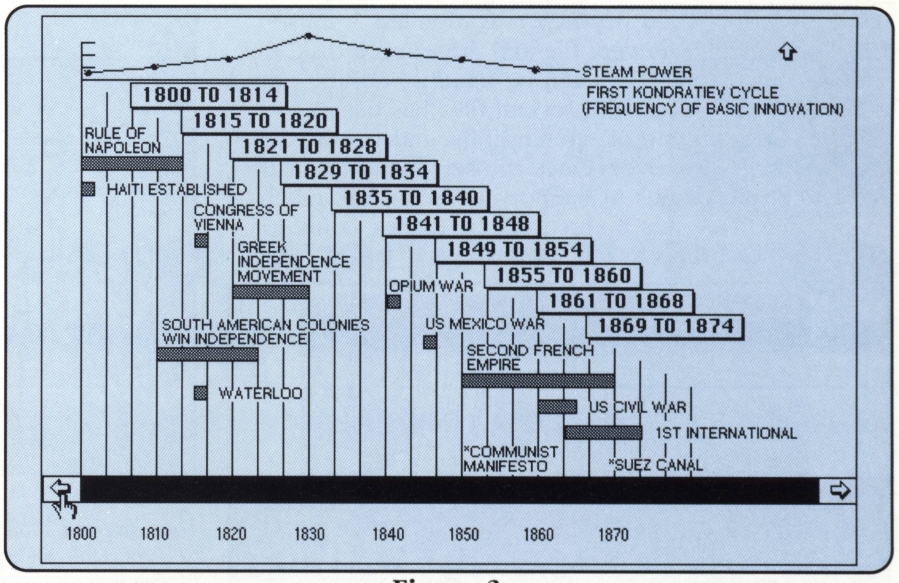


Figure 3

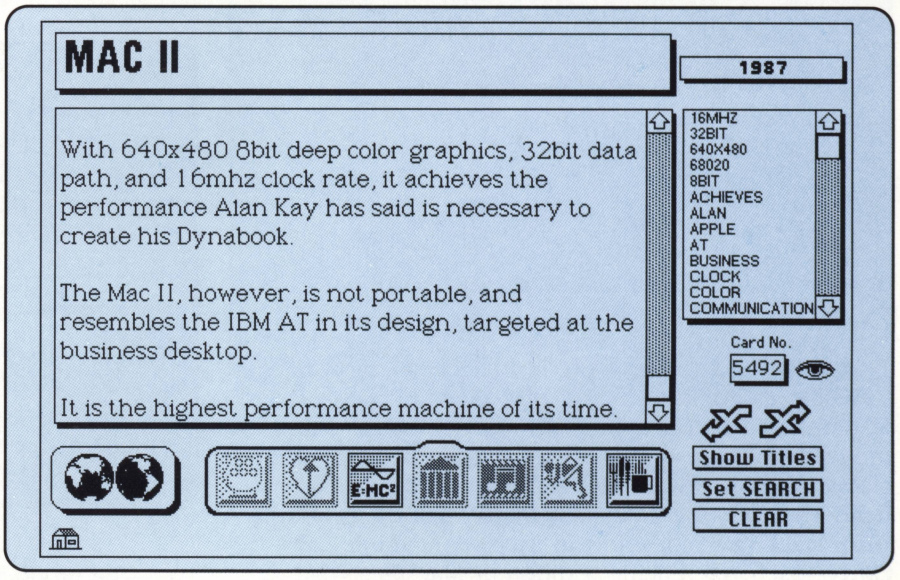


Figure 4

on the actual date takes you to a field that shows all of the events indexed in the Time Table that occurred in that span of years. You then select a name and are taken to the Time Table card for that event.

Xearching for Data

This product uses its own XCMD based search engine (Xearch) to help trace these common events. In the Time Table you use Xearch by clicking on the "Set Search" button at the lower-right corner of the screen. This brings up a dialog that allows for searching not only for single words, but logically "anded" and "ored" combinations of words as well. Plus you can specify a range of dates for the search. Clicking directly on a word in the field to the right searches the entire database for occurrences of the word. The Xearch search is notably faster than similar *HyperCard* searches using the `find` command

This product is expensive, and somewhat limited in nature, but it is not trying to be exhaustive. It is its approach to history that sets it apart from an encyclopedia or a history book, and shows an excellent use of the media. As more modules are developed (the shaded icons across the bottom of Figure 4 give a hint of future modules), it could be used to great advantage by educational institutions to provide an interconnected, non-linear view of history.

HyperExpo in Boston and Beyond...

Reviews of Products

edited by Roger Wood

The first HyperExpo in San Francisco last June was an interesting start for *HyperCard* and hypertext aficionados. Boston HyperExpo in October, although intriguing in terms of new software and some fascinating conferences, seemed a bit sluggish in terms of attendance. Where was everybody?

For all of you who didn't make it, we'll give you a short run-down on some of the products that were shown—and some that weren't.

One important piece of news about which *HyperCard* user should be aware: HyperExpo was not a *HyperCard* only show. Exhibits included Owl International's *Guide*, which runs on both PC's and Macs. (We presented an overview of *Guide* in *HyperLink* Vol. 1, No. 3.) They had some impressive demonstrations of hypertext and multi-media applications such as control of video disks.

In addition, IBM PC only applications like Knowledge Garden's *Knowledge Pro* (also discussed in *HyperLink* Vol. 1, #3) and Cogent Software Limited *HyperBase*, both of which incorporate artificial intelligence with hypertext, showed some excellent capabilities. This important melding of software technologies will have a great effect in the next few years. Millennium Software showed *HyperX*, a *HyperCard* expert systems tutorial that promises a similar melding in the *HyperCard* world. We'll have more about this exciting field in upcoming issues of *HyperLink*, so stay tuned.

ColorCard from Drexel University

HyperCard and hypertext systems have had a great impact in the world of education. Drexel University has several exciting projects using *HyperCard* and related software technologies in the fields of psychology and adult literacy. One of their products demonstrated at HyperExpo in Boston was *ColorCard*.

Douglas Chute of the Neuropsychology department wanted to use *HyperCard* for a project, but needed color to carry it out. *HyperCard* does not support color, but that didn't stop Drexel. They went about developing XCMD's to bring color to *HyperCard*—

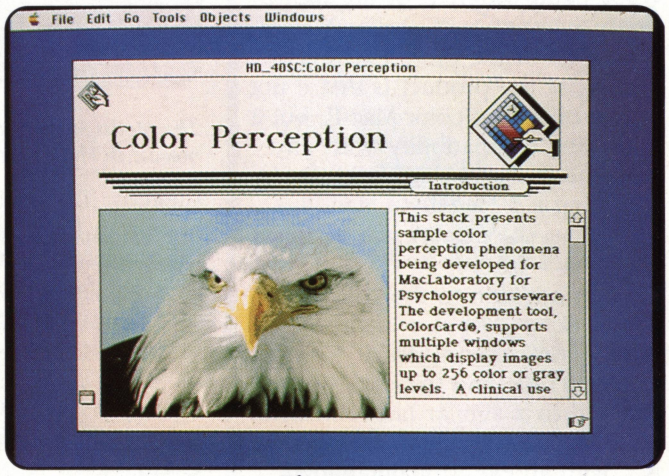


Figure 1

An example of Color Card in action in this card from Drexel's "Color Perception" stack.

Figures 1 and 2 show the results. Although *ColorCard* is still under development and won't be released until 1989, the demonstration was quite impressive.

ColorCard employs a series of XCMD's that create a multi-window environment within *HyperCard*. Windows are created by bringing up the Window dialog

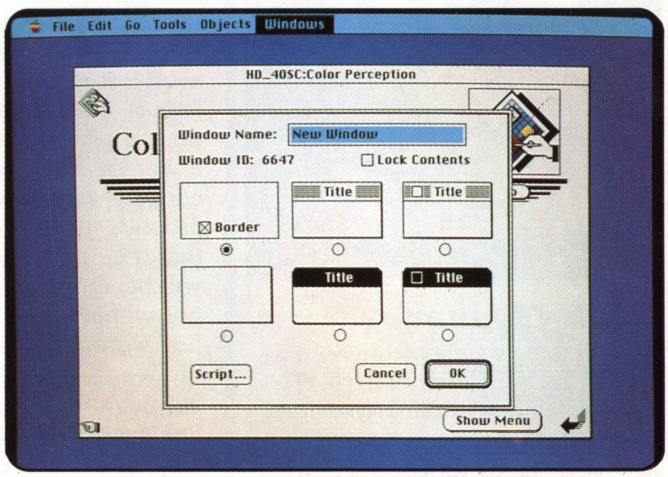


Figure 2

The ColorCard *XCMD's will let users create color windows within* HyperCard *to display drawings imported via the clipboard or scrapbook from color drawing programs such as* Pixel Paint *or* LaserPaint II, *as well as color scanned art.*

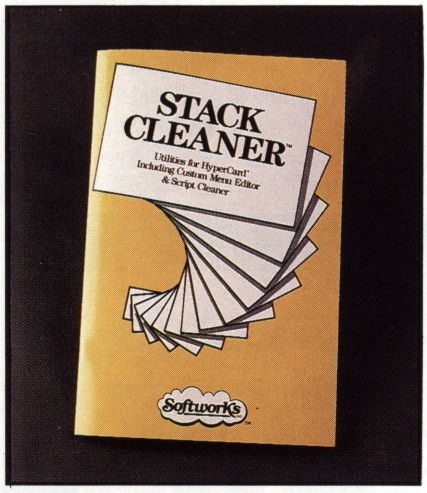
Roger Wood is the Editor of HyperLink Magazine.

box by selecting the option from a Window menu (see Figure 2). The window types are standard Mac windows, with the addition of a borderless window but they can be resized as easily as *HyperCard* fields. Plus, scripts can be associated with the windows so HyperTalk commands can be utilized

You can import any gray scale or color graphics directly into the window employing the scrapbook or the clipboard (up to 256 levels of gray or 256 colors with the 8-bit video board in the Mac II). These graphics must be created using a color paint program or an appropriate scanner.

One of the more exciting aspects of the product is that it not only works on a color Mac II, but it maintains compatibility with any machine that will run *HyperCard*, including a 1 Megabyte Mac Plus. Of course, the display will be black and white.

In addition to the color windows, *ColorCard* also incorporates color visual effects similar to those in HyperTalk, and they will be scripted in a similar fashion. All in all, this is a well thought out product, with an eye to maintaining compatibility with *HyperCard* and the Macintosh interface.



Stack Cleaner

This new utility from SoftWorks, Inc., creators of *HyperTools #1* and *#2*, brings the *HyperCard* user and stackware developer some new very handy tools.

The *Stack Cleaner* name is deceptive. It is actually a collection of six different tools and is named primarily for one dubbed the Script Cleaner. To facilitate moving resources from one stack to another the package contains *Rescopy* licensed from Apple Computer. The other features are Split Script, Stack Merge, Clone Stack, and Menu Maker.

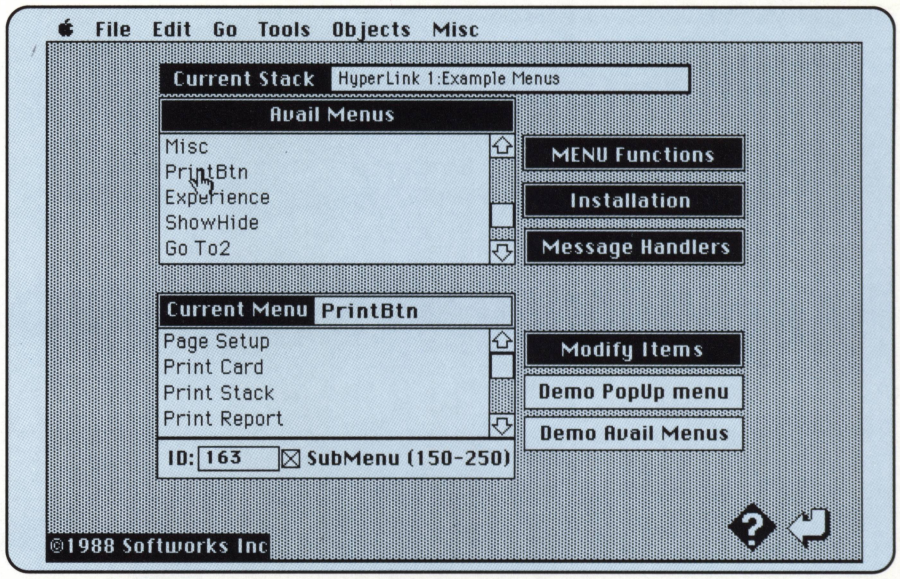


Figure 3

The Menu Maker option of Stack Cleaner *automates the creation of several kinds of menus in* HyperCard.

The Script Cleaner function helps to make stacks smaller by changing all references to *HyperCard* objects to their approved, and smallest, abbreviations. For example, `background` becomes `bg` and `field` becomes `fld`. In addition, it keeps complete statistics on the operation, reporting the number of each type of change, and the total amount of stack reduction.

There are a couple of possible drawbacks to using Script Cleaner. the obvious one is readability. If you prefer complete words, even if it means a larger stack, this feature won't appeal to you. In addition, it pays no attention to whether a word is in quotes. If your script contains the words "field," "background," or any of the other words that the Script Cleaner affects these will be shortened even if they are not within commands. There are so few words affected that this probably won't present a problem—but we can conceive of situations where this might affect a stack adversely.

Split Stack and Stack Merge do just what you would expect, automating a process that can be quite tedious when done manually. The Merge function copies only one of the two stack scripts, but this is to avoid `openStack` and `closeStack` handler conflict.

Clone Stack duplicates an existing stack by selectively stripping text out of fields, and by selectively deleting unique cards (i.e., those that have either card buttons or fields).

These are all useful features, but the function that makes this package worth the price is the Menu Maker (see Figure 3). It allows you to create, install, and modify custom menus—both for the Macintosh menu bar, and popup menus. The process itself is automated through popup menus and custom dialog boxes. The only scripting required is if you wish to create or edit message handlers, and even this is aided through the "Message Handlers" popup menu in Figure 3.

The only part of the process that a user might find intimidating is when asked to select a unique menu ID#. Even here you are prompted to select a number in an appropriate range. After a little practice this too is pretty easy.

This is also the first *HyperCard* menu extension to support the easy creation of submenus—the menus that pop out of the side of an ordinary menu with further options. These are a relatively recent addition to the Macintosh interface. They will be familiar to those who have used *PageMaker* 3.0 or *Full Write Professional.*

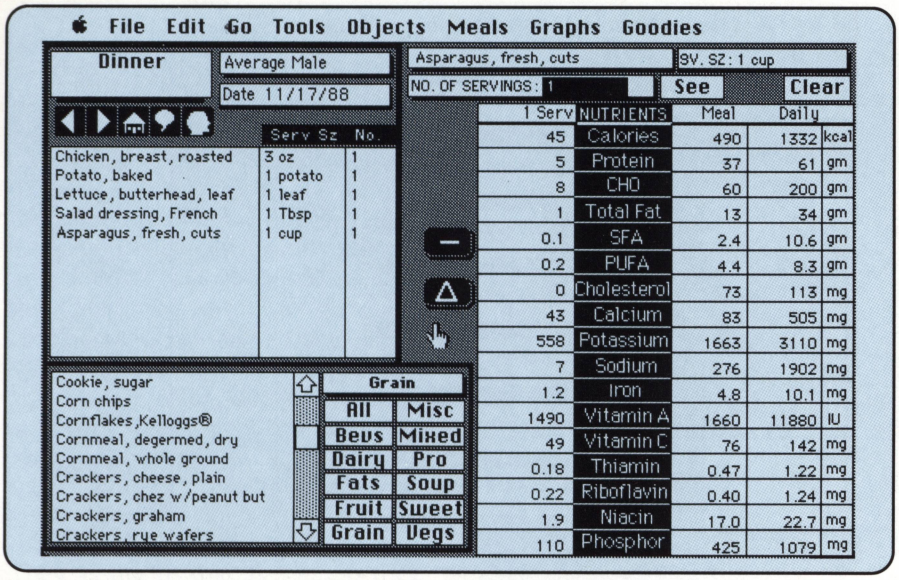
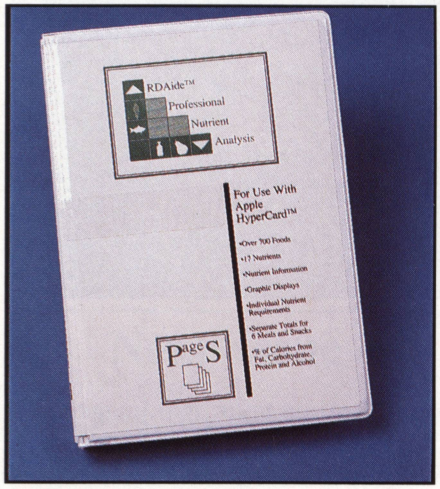


Figure 4

The meal entry screen from RDAide *is quite functional, but a bit cluttered.*

By the way, Softworks has made a deinstall of *HyperTools* available on CompuServe, AppleLink, and the Source under the file name "HT Deinstall." This was a limitaion in the product mentioned in last issue's review. We're very pleased that SoftWorks was so receptive to user demand.



RDAide

These days we're all increasingly interested in diet and fitness. Print and Graphics Educational Systems recently released the *RDAide HyperCard* stack that is designed to aid in tracking your diet in a useful way. It not only provides the nutritional content of what you eat, but it offers guidence to help tailor your diet to your own personal dietery needs.

When you begin, you enter personal data such as sex, age, weight, height, level of activity, and any goals such as weight loss or gain. These can be quite specific. Women can even enter facts such as being pregnant or lactating to help guide the stack in giving feedback.

Next you are taken to a chart like the one in Figure 4. Notice the three added menus: Meals, Graphs, and Goodies (created using the Menu XFCN's and XCMD's by Michael Long of Nine to Five Software). If there is any complaint that we have with this stack, it is with this card. It is cluttered and was, at first, a bit intimidating. After using the stack, however, we found that it was functionally sound.

You enter a full day's meals by selecting each meal from the Meals menu. These are breakfast, lunch, dinner, and three snacks so you can be very accurate in your input. Foods are selected by clicking on each category of food listed in buttons at the bottom center of the screen. This brings up specific items in the scrolling field at the lower-left corner. We found all the foods we required to describe our diet listed in this database. If you need to, there is a relatively easy method of adding foods under the Goodies menu. The only difficulty is that you might have to do a little study to give the program the kind of complete information that is available about the foods already there.

After you enter all the meals, and this only takes a few minutes, you can see how you're doing by choosing one of the options from the Graphs menu. One is a pie chart depicting the percentage of protein, fat, and carbohydrates in the day's calorie intake. Another graph is a bar chart that demonstrates how well your diet measures up in comparison withthe recommended daily allowance for eight different nutrients (see Figure 5). As this figure shows, the information is very complete.

The third item in the Graphs menu provides feedback on your dietary goals based upon the information you put into personal information card. These items take the form of goals versus actual amounts in terms of calories, total fat, poly-unsaturated fat to saturated fat ratio (P/S Ratio), cholesterol, sodium, and calcium to phosphorus ratio. If these sound a bit complicated, they're not. The stack does all the difficult figuring and gives a simple "yes" or "no" as to whether the dietary goals for the day were met based upon your input in the personal information system.

In addition, the information for the day can be printed out selectively or all of it, with no need for a lot of complicated report formatting.

A lot of care and work went into this package and it shows. It was carefully crafted by people who know the subject. We realize that other programs have appeared on other computers which claimed to do what this one does. When we first looked at it, we were sceptical—but this is a great tool for tracking your diet. It makes use of *HyperCard* to bring good dietary advice to the user by simply taking in data and giving useful feedback.

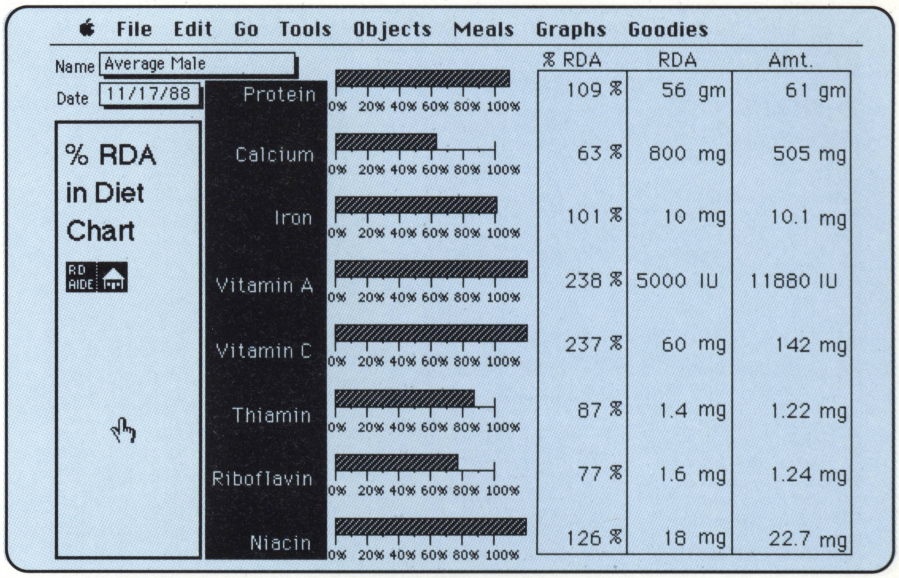


Figure 5

The RDAide *RDA chart gives complete feedback on how well your diet fulfilled the recommended daily allowance for eight important nutrients based upon the information you provided on the personal information card.*

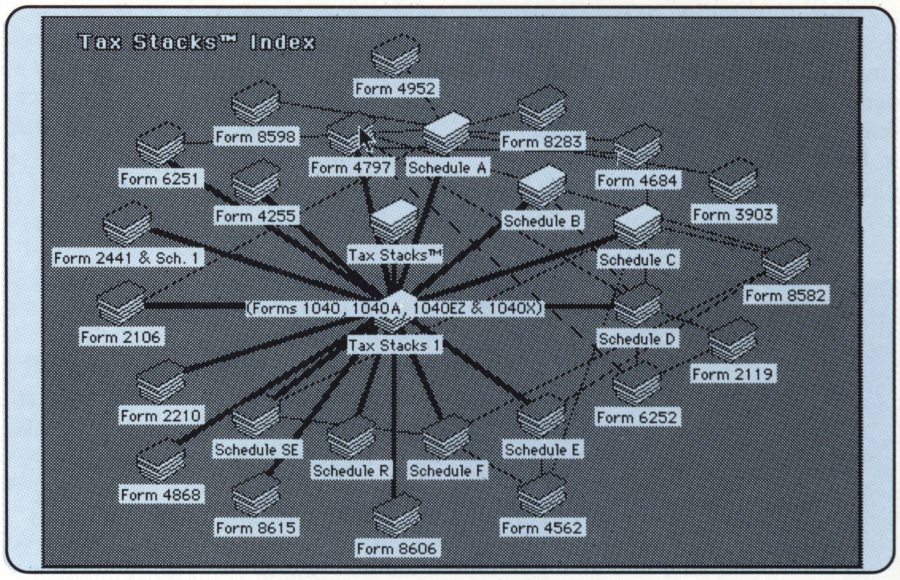


Figure 6

Tax Stacks

As the new year dawn's we look back on last year and realize—it's time to do the taxes. Not a pleasing thought... where will we get help year? How about *HyperCard?*

StackWorks is in the final stages of putting together this years Tax Stacks. This stack asks a series of questions—most are all too familiar—and you just check the boxes and fill in some blanks. But, instead of having to figure out which forms you need, the stack does it for you, leading you to the correct form based upon your answers. It lets you copy the information from your W-2 form right into the computer, and moves it all to the appropriate boxes on the form. If you want to itemize deductions, up comes schedule A for you to fill in. In fact *Tax Stacks* plans to have all the forms a private citizen and many small businesses will ever need (see Figure 6). These won't just be close to the real thing, they will be 100 percent IRS approved. You can print them on your own ImageWriter or a LaserWriter—even a Linotronic if you have around.

If there is a piece of information you can't locate, but don't want to stop, you click on a small flag marking the card where the missing info can be entered at a later time.

There's an on-line calculator to do the figuring, and it keeps a scrolling "tape" of all your calculations for later perusal. There's also a small notebook for keeping notes on things you may want to check on later. And, if you get bored, or disgusted with the tax situation, *Tax*

Stacks has on-line humor in the form of tax jokes like: "You've got to hand it to the IRS—If you don't, they will get it anyway."—Ralph L. Woods.

After you've entered everything, the forms are displayed on the screen for you to double check. Then you click the "Print" button, and your IRS approved forms are generated. Sound too good to be true?

There is, at this writing, one small hitch in this otherwise blissful scenario. StackWorks has not released the final stack. Why? IRS hasn't handed down the last edict yet in order for StackWorks to do the math for the final tax figures. They assure us that as soon as they get the word from IRS, they will be able to implement the final version. When asked for a release date, Sandra Levin of StackWorks just smiled. "To a great extent," she said, "it's up to IRS." From what we could find out from StackWorks, it should be soon after the first of the year. Until then a demo is available from StackWorks, and they are accepting orders.

The demo we saw appeared to work as advertised, with certain messages like, "This form not in this demo version, but will appear in Tax Stacks." We'll keep you posted on the progress, or you can contact StackWorks directly.

- **For information regarding pricing and availability of products reviewed above see the next page.**

Next Issue

Some very exciting and important developments are on the horizon in *HyperCard*. A number of stacks are in late beta as we go to press, and we will look at several of them in the next issue. Look for reviews of:

- *Focal Point II* from Mediagenic
- *Organizer + 1.5* from Dazzl
- *Contacts!* from Applied Imagination
- *Client* from SoftWorks
- *Elaborate* from Mind Vision and more...

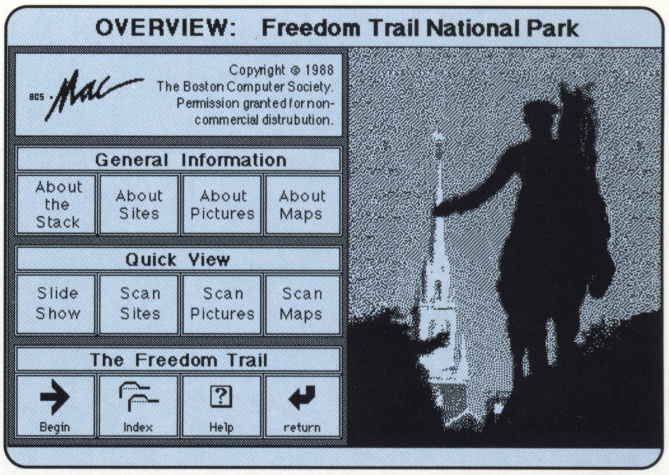
The Boston Computer Society Challenge

The Boston Computer Society - Macintosh (BCS-Mac) showed a *HyperCard* stack at HyperExpo in Boston which is a guided tour of the Boston Freedom Trail National Park. It is a nice combination of scanned graphics, maps, and text that will teach even a Beantown die-hard a thing or two about the famous Freedom Trail.

BCS Mac would like to challenge other user groups around the country to try to outdo them with stacks showing local sights and things to do in their own towns.

To get a copy of this disk send $10 ($4 for BCS members) plus $3 shipping to:

BCS Mac
48 Grove Street
Somerville, MA 02144
(617) 625-7080



Guide to Reviewed Products

Product Name: *ColorCard*
Company Info:
Drexel University
Office of Computer Services
Philadelphia, PA 19104
(215) 895-2667
Price: $28
(Not available until Early 1989)
Developer: Drexel University

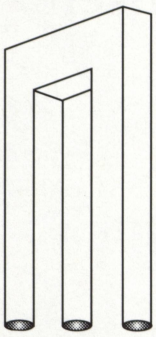
Product Name: *Stack Cleaner*
Company Info:
Softworks Inc.
P.O. Box 2285
Huntington, CT 06484
(203) 926-1116
Price: $50
Developer: John DiBiasi

Product Name: *RDAide*
Company Info:
Print and Graphics Educational Systems
2070 30th Ave
San Francisco, CA 94116
(415) 665-3924
Price: $85 + S/H
Developer: Glenn G. Lew, DMD
Patricia Booth, MS, RD

Product Name: *Tax Stacks*
Company Info:
StackWorks
P.O. Box 426
Urbana, IL 61801
(217) 328-5257
Price: $70 + S/H
(Not available until Early 1989)
Developer: Sandra Levin

MARKETLINKS

ManorTools™
"THE HyperCard Toolkit"



Still $59.95 Retail!

Still THE Best!

NOT Another Illusion!

- Icon Editing
- Cursor Editing
- Button/Field Alignment
- Sound Importing/Exporting
- SoundWave™ Compatible
- SoundEdit™ Compatible
- XCMD's/XFCN's
- Sound/Icon Library
- and much, much more!

MasterCard/Visa/Discover
Call (800) 876-2300
(24 Hour-7 Day Order Line!)

Or, save shipping charges by sending check for $59.95 to:
Manor of Micro, Inc.
6827 Black Duck Circle
Lino Lakes, MN 55014

Dealers May Order From PC Micro Dealers Coop at (800) 448-8892.

INTERACTIVE DESIGN

Custom HyperCard solutions for business. We've built stacks for Apple Computer, US West Direct, MacZone, and authored *"101 Scripts and Buttons for HyperCard"*

206-542-9000

HYPERCARD AUTHORING SERVICE

Realize the full potential of HyperCard by converting your books, catalogues and files into **HYPERTEXT** Documents. Professional editor/writer/proofreader will organize and tailor your information into custom stacks. For more information call:

COOL FIRE TECHNOLOGY
133 West 15th Street, Suite 12
New York, N.Y. 10011

(212) 807-0513

HyperCard Bible
New King James Version **With:**

• SuperSearch™

$95.00 Visa, MC accepted

Wayzata Technology
5410 Three Points Blvd, Suite 435
Mound, Minnesota 55364
(612) 472-4836

MIDI Mac-Xchange

MAC MIDI & UTILITIES

We Sell, Buy, Trade, License and Co-distribute your MIDI Stackware & Utilities. Send $5.00 (refunded w/purchase) for catalog of inexpensive MIDI-ware. Generic & Hardware Specific.

THOMAS CONSULTING SERVICES
1328 Chestnut Street Suite 350, Emmaus, PA 18049

Where's that article?

Hyper Mac Subjects™
is a Subject/Name Index of articles, letters, reviews, tips, etc. in **20** or more Macintosh magazines - in 1 stack
$50/yr
($US60/yr foreign orders)
for 6 issues published every 2 months by:
Pointer Publications,
935 Brooksglen Dr., Roswell GA 30075.
(404) 587-1844 (check/money order only)

A Portable Mac!
The WalkMac SE

12 lb Portable
8 MBytes of RAM
Battery Capable
BackLit LCD Display
Accelerator/Math Co-processor

COLBY SYSTEMS CORP.
(209) 222-4985
also new from Colby . . .
Mac SCSI 400 MByte WORM Drive

MARKETLINKS Order Form

Company Name _____
Contact Name _____
Address _____
City _____ **State** _____ **Zip** _____
Area Code _____ **Phone** _____

Yes, we want to take advantage of HyperLink Magazine's special MarketLink Advertising Spaces.
We have marked the desired space below. *Enclosed is our ad copy on disk and an actual size hardcopy.* Also enclosed is our payment in full for this ad insertion.

A	B	B	C
$88	$176	$176	$352

We understand that our ad copy will be typeset by LaserWriter using the closest fonts to those in our ad copy as available. I agree that HyperLink's liability in running this ad shall not exceed the cost of the ad.

Visa or M/C # _____ **Exp.** _____
Signed _____ **Date** _____

New LINOTRONIC Service Bureau

No-sales-tax Oregon has a new Linotype Linotronic Laser Imagesetting service. True high resolution pages from Macintosh (PageMaker, R.S.G., Quark, etc.) or IBM and compatibles (Ventura, etc.), disk or modem.

Professional Design & Typography for Self-Publishers, Advertising Agencies, and Printers in Oregon since 1979

Editing & Design Services Inc.
30 East 13th Ave. / Eugene / 683-2657
FAX: 503-683-6184

Send your ad and payment to:
MARKETLINKS
HyperLink Magazine
P.O. Box 7723
Eugene, OR 97401

Questions?
Call Carole
1 (503) 484-5157

Order NOW!

1 800 544-0339
Please send me the free

StackSampler

Jan/Feb 1989 Disk
with sample stacks, demos, and information from HyperLink Magazine's Advertisers.

I understand there is a shipping and handling fee of **$2.95**.

Enclosed is a check or charge my Visa/MasterCard #

_____

Expiration Date _____

Signed _____

Name _____
Address _____
_____
City _____ State ____
Zip _____
Phone (___)_____

StackSampler
from Our Advertisers


Applied Imagination
AI Showcase


Heizer Stack
Exchange


HyperAnnex
Sampler


Manor of Micro, Inc.
SoundRes Demo


PageS
RDAide Demo


Softworks, Inc.
HyperTool Demos


StackWorks, Inc.
Tax Stacks Demo

StackSolutions Disk Directory


Keyword in Context 1.8


Shafer On Scripting V2,#1


TEX by ^z - v.0.51


HyperCard Tips V2,#1


Personal Financial Statement


Xpanding HyperCard V2,#1


HyperCard Haiku V2,#1


InventoryStack.update

Stack	Pg.
Keyword in Context	16
Shafer On Scripting	35
TEX by ^z - v.0.51	35
HyperCard Tips	38
Personal Financial Stmnt.	39
Xpanding HyperCard	43
HyperCard Haiku	46
InventoryStack.update	n/a

Stack Update

As we went to press we learned of some small bugs in last issue's "InventoryStack," so we are including an updated version on this issue's *StackSolutions* Disk.

HYPERLINK MAGAZINE

The premier HyperCard information magazine! Subscribe Now!

When you subscribe to **HyperLink Magazine**, take advantage of our special offer to receive the **StackSolutions** microdisk for your first subscription issue — **FREE!** *
Subscribe TODAY! Mail this subscription form and payment to:

HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401.

In a hurry? Place your order by calling our toll-free subscription line at 1-800 544-0339. Please have your VISA or MasterCard handy.

SUBSCRIBER NAME _____ PHONE (____) _____
COMPANY _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____

☐ This is a gift subscription for the person listed above. My name and payment choice is listed below.

GIFT DONOR NAME _____ PHONE (____) _____
COMPANY _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____

METHOD OF PAYMENT: ☐ VISA ☐ MasterCard ☐ Check (enclosed)
Credit Card Number _____ Credit Card Expiration Code _____

Signature of Card Holder _____

CHOOSE one of the following HyperLink Magazine subscriptions —

☐ One Year $25 ☐ Two Years $45 ☐ Three Years $60 (HyperLink is a bi-monthly publication)

☐ Yes, send the free StackSolutions disk. Enclosed is the $2.95 Shipping & Handling charge.*

OR CHOOSE a combination HyperLink Magazine/StackSolutions Disk subscription —

☐ One Year $67.95 ☐ Two Years $124.95 ☐ Three Years $172.95
(The combination subscription prices include $2.95 S&H for the first disk, which is free.)

ALSO PLEASE SEND THE FOLLOWING PRODUCTS—

Please send the HyperLink Magazine back issues marked.
I enclose $4.95 for each one marked.

☐ Vol.1 #1 ☐ Vol.1 #2 ☐ Vol.1 #3 ☐ Vol.1 #4

Please send the StackSolutions back issue disks marked.
I enclose $14.95 (includes S&H) for each one marked.

☐ Vol.1 #1 ☐ Vol.1 #2 ☐ Vol.1 #3 ☐ Vol.1 #4

☐ Vol. 2 #1 (for this issue)

All orders must be in US funds.

Canada/Mexico, added air shipping per year

Magazine only	Magazine and Disk
$15	$21

Foreign orders, added air shipping per year

Magazine only	Magazine and Disk
$36	$54

Total amount enclosed for the checked items $ _____

*Please note that the premium (StackSolutions microdisk) has a retail value of $12. If for any reason a refund is to be made on this subscription, the value of the premium and any delivered issues are subtracted from the refund. Also, please note: The $2.95 Shipping & Handling must be paid with your magazine subscription to take advantage of this offer.

STORAGE

*** Limited time special offer!**
Up to $300 cash for your old hard disk drive when you purchase a DPI 44R.

$1695	$499	$799	$1249	$1399	$1099	$1249
DPI 44 Removable	***DPI 20 External***	***DPI 60 External***	***DPI 100 External***	***DPI 144 External***	***DPI 100 II Internal***	***DPI 144 II Internal***
25ms, 10 Megabits/Sec. data transfer rate, shock resistant to 30Gs.	65ms, 7.5 Megabits/Sec. data transfer rate, shock resistant to 50Gs.	48ms, 7.5 Megabits/Sec. data transfer rate, shock resistant to 50Gs.	28ms, 10 Megabits/Sec. data transfer rate, shock resistant to 50Gs.	28ms, 10 Megabits/Sec. data transfer rate, shock resistant to 50Gs.	28 ms, 10 Megabits/Sec. data transfer rate, shock resistant to 50Gs.	28ms, 10 Megabits/Sec. data transfer rate, shock resistant to 50Gs.

Unlimited capacity on a fixed budget.

Our 44MB removable hard disk drive is not only fast (25ms average access time), it's also extremely flexible. With the removable Data Cartridge, you're assured flexibility, portability, security, and an unlimited amount of storage space.

Application specific.
The DPI 44 removable is designed with the demanding user in mind. Whether you're working with spreadsheets, desktop publishing, graphics, or multiple data bases, the 44R has both the speed, and unlimited storage capacity.

Goes anywhere.
Whether you need to take information with you, or send it by courier, the removable Data Cartridge is ideal. Rugged (tested to 30G's), and lightweight enough you won't lose time and money shipping cross-country.

Complete data security.
What could be safer than being able to take your data with you? No more unauthorized access by outside system users. The cartridge is small enough to fit inside any safe, briefcase, or desk drawer.

Unlimited storage.
You simply need to buy additional cartridges. The storage possibilities are endless. You could back-up your hard disk drives in less time than conventional tape drives, and without the possibility of data loss due to tape-stretch.

****Limited time special offer!***
With the purchase of a DPI 44R, you have the opportunity to trade-in your existing hard drive for cash.. Yes, we'll buy your old hard disks, working or not. You could earn up to $300 cash depending on the drive you trade in. Call for details on how you qualify to save big bucks.

A family tradition.
In addition to our DPI 44R, we also manufacture both Internal and External hard disks. From our Internal 100 and 144MB, to our External 20, 60, 100, and 144MB drives, you can count on DPI's service and guarantee. A full 30 day money-back and one year on both parts and labor. No matter what the reason, DPI will repair or replace your drive, absolutely free!

What are you waiting for? Add a DPI hard drive to your system today and come out of the dark ages.

DPI

40 Corning Avenue, Milpitas, Ca 95035
408/945-1850 800/825-1850

VISA MasterCard

Prices quoted are for cash purchases. California residents add 7% sales tax. Price does not include system cable or daisy chaining cable nor shipping. DPI 44 Removable, 20, 60, 100, 144, 100 II, and 144 II are trademarks of DPI. Macintosh is a registered trademark of Apple Computer. Prices subject to change without notice.